Unsupervised Context-Driven Recommendations Based On User Reviews

Francisco J. Peña
Insight-Centre for Data Analytics
Department of Computer Science
University College Cork
Cork, Ireland
francisco.pena@insight-centre.org

ABSTRACT

In this work we present Rich-Context, a context-driven recommender system that extracts contextual information using topic modeling without the need to define keywords. Our system uses the mined context to produce recommendations. We propose a methodology to measure the quality of context topic models along with a novel way to represent context that allows it to be used as side-information in a recommendation engine. Results show that Rich-Context makes more accurate predictions than five well-established recommendation algorithms.

CCS CONCEPTS

• **Information systems → Collaborative filtering**; **Recommender systems**; • **Computing methodologies → Topic modeling**; *Natural language processing*; *Unsupervised learning*;

KEYWORDS

Context-driven recommender system, context-aware recommender systems, topic modeling, review-mining

ACM Reference format:
Francisco J. Peña. 2017. Unsupervised Context-Driven Recommendations Based On User Reviews. In *Proceedings of RecSys '17, Como, Italy, August 27-31, 2017,* 5 pages.
https://doi.org/10.1145/3109859.3109865

1 INTRODUCTION

User reviews have become a source of additional information for recommender systems to work with besides ratings. Many recommender systems have successfully extracted information from user reviews in order to make better recommendations ([6]).

Additionally, [1] has also shown that including contextual information into recommender systems leads to improved recommendations. Since context has an influence on how a user perceives a product or service, the same user could rate the same item with totally different (1 to 5) ratings depending on the context in which the item is consumed. For instance, if a user is on a business trip, she could rate a small hotel room with 5 stars because she is mainly concerned about the quality of the Wi-Fi, whereas if she goes to the same hotel with her husband and kids, she could rate it with 2 stars because of the room size and the lack of a swimming pool for the children. When users rate services, they are not just rating the facilities, but their overall experience.

Approaches such as [11, 17, 18] have successfully incorporated contextual information into their recommendation models to produce more accurate recommendations. They have used datasets that contain explicit information about the context in which the items were consumed. The problem with many real-world systems is that contextual information is not often available or it is very limited. This happens because users are often not bothered about stating the context in which they visited a restaurant or hotel, and web portals normally limit what they ask users for to a couple of contextual features (typically the purpose of the trip/meal and the companion). To overcome this lack of extra contextual information, several context-aware recommender systems have started to incorporate contextual information from user-generated reviews in order to improve the recommendation models ([5, 10]).

When users write reviews, they tend to do it in two styles: giving detailed stories about their visit, or writing general overviews of a place. Detailed stories are told when users are describing past experiences in a hotel or restaurant. On the other hand, most of the time general overviews cannot be related to a particular visit in the past. Since every experience happens under a context (there is no such thing as a context-less experience), we assume that reviews that describe experiences are going to contain more contextual information than the ones that do not describe experiences. We call reviews that describe experiences *specific* and the ones that give general descriptions *generic* ([2]). Reviews in which the author is describing an experience with a product or service are specific; for instance: *"During the summer, we like to take a mini staycation. This year it was extra special as we also got engaged. Our stay at the Biltmore was just fantastic. The service was exceptional, and the food was amazing"*. Generic reviews just give general opinions and do not describe an experience with the product or service; for instance: *"Nice hotel, all the amenities you need, great complex of pools"*.

In this paper we present Rich-Context, a recommender system that extracts contextual information from user reviews using unsupervised topic modeling algorithms. This has the benefit of not having to predefine keywords that describe context, capturing contextual situations that were not considered before, such as wheelchair accessibility, parking facilities or pet friendliness, we call this *latent*

RecSys '17, August 27-31, 2017, Como, Italy

ACM ISBN 978-1-4503-4652-8/17/08…$15.00
https://doi.org/10.1145/3109859.3109865

context. We introduce a new way to measure the quality of topic models in terms of their context-richness and present a novel way to represent contextual information. Tests on multiple datasets show that our system can beat several other well-known recommenders in terms of rating prediction.

The remainder of this paper is organized as follows: Section 2 discusses related work, followed in Section 3 by the methodology to create a Context-Driven Recommender System. Section 4 introduces a series of metrics to measure the quality of topic models in terms of context-richness. Section 5 presents the results and Section 6 contains the conclusions and the future work.

2 RELATED WORK

Our work relies heavily in two areas, review-based recommender systems and topic modeling. Latent Dirichlet Allocation ([4, 9, 16]) is arguably the most popular topic modeling algorithm. LDA is a generative probabilistic model in which documents are represented as random mixtures of latent topics, and each topic is characterized by a distribution over words. The main problem with LDA is that due to its stochastic nature, it produces different topic models on separate runs over the same data when using distinct random seeds. To overcome this issue, [3] proposed a technique called Ensemble Topic Modeling that is based on Non-negative Matrix Factorization and it factorizes a stacked matrix composed of several runs of the same topic model using different random seeds. Results show that it reduces variability, producing more stable and reliable topic models.

[12] link LDA with matrix factorization via a transformation function to learn more accurately the parameters that lead to better recommendations. [8] presents an unsupervised model that mines aspects from movie reviews using topic modeling and integrates the mined aspects into the recommendation engine. [5] uses heuristics to extract contextual information from reviews along with item aspects and sentiment. [10] uses a supervised topic modeling method (Labeled-LDA) to classify reviews based on their context. These last two methods are the most similar to ours because they also extract contextual information from user-generated reviews. The problem with these methods is that all of them have to predefine the possible types of contextual variables and their values, thus leaving out many other contextual possibilities.

[2] presents a method to extract contextual information from user-generated reviews through topic modeling and word similarities. We borrow some of their ideas to build our recommender system, such as the separation of reviews into specific and generic, and the use of topic modeling to discover contextual information.

3 METHODOLOGY

Rich-Context is a query-based context-driven recommender system ([13]). In our system the users write queries stating the context in which they intend to travel and then receive a list of recommendations. It has three main components: RCClassifier, RCMiner and RCRecommender. RCClassifier separates reviews into specific and generic, RCMiner creates topic models using only specific reviews and then labels topics as contextual or non-contextual. Finally, RCRecommender applies the topic model to the input query and uses the contextual information contained in the query as side-information to produce recommendations. An example of a query for hotels can be *"summer holidays children"*. Here the user is indicating the context under she wants to travel.

3.1 Classifying Reviews

To classify reviews we first created a training set of manually labeled reviews. We manually labeled 300 reviews from each of our datasets, assigning two classes: *specific* or *generic*. Reviews that told particular experiences during a visit were labeled as specific and reviews that gave a general overview of a place were labeled as generic ([2]).

We perform part-of-speech tagging on those reviews and then give them to a logistic regression classifier. The features we use in our classifier are a subset of the features specified in [2] that, after running several tests, helped us achieve the best classification performance, viz:

- *LogWords*: log of number of words in the sentence + 1
- *Vsum*: log of number of verbs in the sentence + 1
- *VBDSum*: log of number of verbs in the past tense in the sentence + 1
- *ProRatio*: ratio of log of number of personal pronouns + 1 to *LogWords*

3.2 Creating Contextual Topic Models

To obtain contextual information from user-generated reviews, we use topic modeling. The basic idea behind this is to generate a topic model from a set of documents and then distinguish between topics that contain contextual information and topics that do not.

In topic modeling documents are expressed as a mixture of topics. Algorithms such as Latent Dirichlet Allocation take a probabilistic view of topic modeling, in which the documents are represented as a probability vector of topics. Other authors have used Non-negative Matrix Factorization (NMF) for topic modeling. Using NMF we have a document-term matrix $\mathbf{A} \in \mathbb{R}^{d \times t}$, where d is the number of documents and t is the number of terms. Matrix $\mathbf{A}$ can be factorized into two matrices $\mathbf{H} \in \mathbb{R}^{d \times k}$ and $\mathbf{W} \in \mathbb{R}^{k \times t}$ so that $\mathbf{A} \approx \mathbf{WH}$. The $\mathbf{H}$ matrix may be seen as a document-topic matrix and the $\mathbf{W}$ matrix can be seen as the topic-term matrix.

To create the topic models, we use the methodology of ensemble topic modeling proposed by [3]. The goal of this methodology is to reduce the variability that is characteristic of topic models built using stochastic algorithms. Since stochastic methods are used, two topic models that are created using the same dataset but different random seeds can be quite different. The goal of ensemble topic modeling is to reduce that variability. This is done by creating a set of base topic models by executing NMF r times applied to the same document-term matrix $\mathbf{A}$, then the r generated matrices are stacked together in one matrix and a final run of NMF is applied to the stacked matrix in order to obtain the ensembled topic model ([3]).

After we have classified reviews as specific or generic using RCClassifier, RCMiner creates a topic model using the ensemble topic model approach proposed by [3] using only specific reviews. Once the topic model has been created, we apply the topic model to all the reviews (specific and generic) to obtain the vector of topic weights for each review and then normalize the vector so its sum adds up to 1.0. Then, we calculate which topics appear more frequently in specific reviews than in generic ones. We do so by

using a variation of the approach proposed in [2] in which, instead of computing cardinalities, we sum weights for each topic. We say that topics that appear more frequently in specific reviews than in generic ones, are contextual topics and those that appear more in generic reviews are non-contextual topics.

$$w^s(t_k) = \frac{\sum_{S \in specific} T[k]}{|S : S \in specific|} \quad (1)$$

We make a similar calculation for generic reviews:

$$w^g(t_k) = \frac{\sum_{S \in generic} T[k]}{|S : S \in generic|} \quad (2)$$

Finally, the likely contextual topics, CT, are those where the ratio of the two proportions exceeds a threshold β:

$$CT = \left\{ t_k \mid \frac{w^s(t_k)}{w^g(t_k)} > \beta \right\} \quad (3)$$

The topic models are created using only the nouns of the specific reviews, as these capture most of the contextual information. As we will show in the results sections, using only nouns translates into more context-rich topic models compared to using words from all part-of-speech.

3.3 Making Context-driven Recommendations

Once we have created the topic models and separated contextual topics from non-contextual topics, we transform our original dataset of records from $\langle\langle u, i, R_{ui}, \rangle, r_{ui}\rangle$ into records where the contextual information is represented by a vector of contextual topics weights $\langle\langle u, i, c_1, \ldots, c_{|CT|}\rangle, r_{ui}\rangle$. We do this by applying the topic model to the review, which will give us a vector with the weights for each topic; we normalize that vector so all the weights add up to 1.0 and then we discard the weights of all the topics that have been marked as non-contextual. We use the vector of contextual topic weights as side-information and include it into the recommendation model. For the recommendations we use the LibFM implementation of Factorization Machines algorithm ([14]). Since we are assuming a context-driven approach, the user supplies a query containing her intended context, similarly represented as a vector of weights.

4 TOPIC MODEL METRICS

To evaluate how good our model is, we measured the quality of the topic models produced. Given that there is not a well-established methodology to measure the quality of topic models in terms of their context-richness that we are aware of, we designed our own metrics. We came up with three metrics to do this: the topic score, the topic model score and the separation score. We also had to manually create a vocabulary of contextual words to help us see if a topic was composed of many contextual words or not. This vocabulary was used only in this evaluation: it is not used in the RC system.

4.1 Topic Score

The topic score measures how rich is a topic in terms of context and is given by $ts(t) = \sum_{v \in V} x_{vt} \cdot b_{vt}$, where x_{vt} denotes the weight that the word v has in topic t, V is the vocabulary of all words, and b_{vt} is a binary variable that indicates if the word v belongs to the manually defined vocabulary of contextual words.

Dataset	Reviews	Users	Items	Sparsity
Yelp Hotels	4085	3408	102	0.988
Yelp Restaurants	148161	35054	2552	0.998

Table 1: Description of the datasets.

4.2 Topic Model Score

The topic model score is just the average of all the topic scores in the topic model, and is given by $tms = \frac{\sum_{t \in T} ts(t)}{|T|}$, where T is the set of all topics in the topic model.

4.3 Separation Score

The separation score measures how well our algorithm separates context-rich topics from non-context-rich topics. It is the average score of the context-rich topics multiplied by a constant γ plus the average score of the non-context-rich topics multiplied by $1 - \gamma$:

$$ss = \left[\gamma \cdot \frac{\sum_{c \in CT} ts(c)}{|CT|} + (1 - \gamma) \cdot \left(1 - \frac{\sum_{n \in N} ts(n)}{|NCT|} \right) \right] \quad (4)$$

where CT is the set of the topics marked as contextual by RCMiner and NCT is the set of the topics marked as non-contextual ($NCT = T \setminus CT$) and γ is a weight that balances the importance of getting high scores on context topics versus getting low scores on non-context topics.

5 RESULTS

To evaluate RCMiner, we used the hotels and restaurants dataset from Yelp that were provided for the RecSys 2013 competition. We removed records that contained items that had less than 10 reviews. Table 1 contains a description of the data after the records were removed.

5.1 Topic Model Quality

In Table 2 we can see the topic model created from the restaurants dataset with 10 topics. The contextual words are highlighted. Since topics 1 to 5 appear more frequently on specific reviews than in generic ones, they have been labeled by RCMiner as contextual, the remaining are labeled as non-contextual. The first column of the Table 2 shows the ratio of frequency appearance of topics within reviews. For instance Topic 1 appears 2.03 times more in specific reviews than in generic ones. As we can see in Table 2 contextual words appear more frequently in topics with high ratio than in topics with lower ratio.

Table 3 shows that topic models that are richer in contextual words are obtained if topic models are created only with nouns of reviews compared to using all the types of words of reviews. Using only specific reviews leads to context-richer topics in the hotels dataset, but in the restaurants dataset the same does not hold. For separation scores there is not a big difference between using nouns or all types of words. For separation scores we use a value of $\gamma = 0.5$. The baselines in Table 3 are the topic models created from all reviews using all types of words, we can see that there is an improvement on creating topic models with just nouns. Note that for the cases where all reviews are used for building the

Ratio	Word 1	Word 2	Word 3	Word 4	Word 5
2.03	night	dinner	friend	saturday	friday
1.61	lunch	today	day	friend	yesterday
1.34	time	couple	week	minute	hour
1.10	breakfast	morning	sunday	club	day
1.07	review	yelp	experience	star	read
0.99	scottsdale	location	town	experience	tempe
0.93	restaurant	phoenix	area	mexican	week
0.85	chicken	pizza	burger	sandwich	cheese
0.75	place	area	bar	love	home
0.72	food	service	mexican	atmosphere	price

Table 2: Yelp Restaurant topic model (Top-5 words)

Dataset	Part-of-speech	Review Type	Topic Model Score	Separation Score
Yelp Hotel	All	All	0.019	–
Yelp Hotel	All	Specific	0.025	**0.509**
Yelp Hotel	Nouns	All	0.037	–
Yelp Hotel	Nouns	Specific	**0.046**	0.504
Yelp Restaurant	All	All	0.003	–
Yelp Restaurant	All	Specific	0.007	**0.498**
Yelp Restaurant	Nouns	All	**0.019**	–
Yelp Restaurant	Nouns	Specific	0.014	0.494

Table 3: Topic model scores across the different datasets

Algorithm	RMSE	MAE
Rich-Context 50 Topics	**0.964**	**0.762**
Rich-Context 30 Topics	0.966	0.753
Rich-Context 10 Topics	0.967	0.759
Factorization-Machines	0.986	0.771
BiasedMF	1.090	0.847
SlopeOne	1.165	0.909
BPMF	1.322	1.049
NMF	1.489	1.143

Table 4: Rating prediction performance on the Yelp Hotels dataset

Algorithm	RMSE	MAE
Rich-Context 50 Topics	**1.019**	**0.803**
Rich-Context 30 Topics	1.019	0.802
Factorization-Machines	1.047	0.824
Rich-Context 10 Topics	1.048	0.824
SlopeOne	1.278	0.973
BPMF	1.305	1.018
BiasedMF	1.333	1.033
NMF	1.378	1.055

Table 5: Rating prediction performance on the Yelp Restaurants dataset

topic model, the separation score can not be calculated since all the topics would be marked as contextual, having $|NCT| = 0$, which makes Equation 4 fail.

5.2 Recommendations

To evaluate our system and compare it with others, we opted to use the RiVal [15] evaluation framework for recommender systems. Each record in the dataset $\langle\langle u, i, R_{ui}\rangle, r_{ui}\rangle$ is transformed to one in which reviews are represented by their corresponding contextual topic vectors, $\langle\langle u, i, c_1, \ldots, c_{|CT|}\rangle, r_{ui}\rangle$. Then, we split the dataset into training and testing sets. First, we train the recommender using only the records of the training set. Then, we test, in which we simulate that the query the user writes is the review R_{ui}, and measure the RMSE and MAE using only the elements in the test set. We repeat this step 5 times, doing 5-fold cross validation. We compared ourselves against several non-context-aware recommender systems algorithms contained in CARSKit ([19]).

In Tables 4 and 5 we can see the results of comparing Rich-Context against five well known recommender systems. We can see that in both datasets we are able to achieve the best rating prediction. The difference between our approach and the pure factorization machines approach is that ours uses the contextual vector as side information, along with the ratings, whereas the factorization machines just uses the ratings. The implementation is the same, the difference lies in that for factorization machines the dataset is composed of records in the form of $\langle\langle u, i\rangle, r_{ui}\rangle$ and for Rich-context the records are $\langle\langle u, i, c_1, \ldots, c_{|CT|}\rangle, r_{ui}\rangle$.

For evaluating the Top-N performance of RCRecommender we are currently running experiments using a methodology based on [7], with results expected soon.

6 CONCLUSIONS

We have presented Rich-Context, a context-driven recommender system that can mine contextual information from users reviews without the need of keywords. The fact that we do not need to define keywords allows the recommender to have a wider range of contextual situations and make more accurate recommendations. We also introduced new metrics to measure the quality of topic models in terms of context-richness. Tests show that Rich-Context has better performance compared against five other well established recommendation algorithms in terms of rating prediction.

Our immediate plans are to evaluate the Top-N performance of Rich-Context and to compare it against context-aware recommenders from CARSKit. We also want to include sentiment analysis as side-information along with context and test it on other recommendation engines that support side-information besides Factorization Machines. To continue our work in building topic models, we would like to find how to improve topic models in terms of their separation scores. We would also like to use the generated topic models to produce explanations of recommendations. Finally, we would like to explore how our recommender performs in cold-start scenarios.

ACKNOWLEDGMENTS

This work is supported by the Science Foundation Ireland(SFI) under Grant Number SFI/12/RC/2289. I would also like to thank my supervisor Dr. Derek Bridge for his contributions.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, 217–253. https://doi.org/10.1007/978-0-387-85820-3_7

[2] Konstantin Bauman and Alexander Tuzhilin. 2014. Discovering contextual information from user reviews for recommendation purposes. In *1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems*. ACM, Foster City, Silicon Valley, California, USA, 2–9.

[3] Mark Belford, Brian Mac Namee, and Derek Greene. 2016. Ensemble Topic Modeling via Matrix Factorization. In *24th Irish Conference on Artificial Intelligence and Cognitive Science*.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[5] Guanliang Chen and Li Chen. 2015. Augmenting service recommender systems by incorporating contextual opinions from user reviews. *User Modeling and User-Adapted Interaction* 25, 3 (2015), 295–329. https://doi.org/10.1007/s11257-015-9157-3

[6] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154. https://doi.org/10.1007/s11257-015-9155-5

[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 39–46. https://doi.org/10.1145/1864708.1864721

[8] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly Modeling Aspects, Ratings and Sentiments for Movie Recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 193–202. https://doi.org/10.1145/2623330.2623758

[9] T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101, Suppl. 1 (April 2004), 5228–5235.

[10] Negar Hariri, Yong Zheng, Bamshad Mobasher, and Robin Burke. 2011. Context-aware recommendation based on review mining. *Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems.*. http://ceur-ws.org/Vol-756/

[11] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 79–86. https://doi.org/10.1145/1864708.1864727

[12] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 165–172. https://doi.org/10.1145/2507157.2507163

[13] Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. 2016. The Contextual Turn: From Context-Aware to Context-Driven Recommender Systems. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 249–252. https://doi.org/10.1145/2959100.2959136

[14] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages. https://doi.org/10.1145/2168752.2168771

[15] Alan Said and Alejandro Bellogín. 2014. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 129–136. https://doi.org/10.1145/2645710.2645746

[16] Mark Steyvers and Tom Griffiths. 2007. *Probabilistic Topic Models*. Lawrence Erlbaum Associates. http://www.worldcat.org/isbn/1410615340

[17] Yong Zheng, Robin Burke, and Bamshad Mobasher. 2013. Recommendation with Differential Context Weighting. In *User Modeling, Adaptation, and Personalization*, Sandra Carberry, Stephan Weibelzahl, Alessandro Micarelli, and Giovanni Semeraro (Eds.). Lecture Notes in Computer Science, Vol. 7899. Springer Berlin Heidelberg, 152–164. https://doi.org/10.1007/978-3-642-38844-6_13

[18] Yong Zheng, Bamshad Mobasher, and Robin Burke. 2014. CSLIM: Contextual SLIM Recommendation Algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 301–304. https://doi.org/10.1145/2645710.2645756

[19] Yong Zheng, Bamshad Mobasher, and Robin Burke. 2015. Carskit: A Java-Based Context-Aware Recommendation Engine. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, 1668–1671.