Whitepaper: The Universal Recommender

A Recommender System for Complex Semantic Networks

Jérôme Kunegis, Alan Said and Winfried Umbrath
Technische Universität, DAI-Labor, Germany
{kunegis,alan.said,winfried.umbrath}@dai-lab.de

September 2009

Abstract

We describe the Universal Recommender, a semantic recommendation engine that applies to semantic datasets, being independent of a specific data model. The Universal Recommender generalizes domain-specific recommenders such as content-based, collaborative, social, bibliographic, lexicographic and hybrid recommenders. To achieve good recommendation accuracy, several novel machine learning and optimization problems are solved by the Universal Recommender.

1 Introduction

In the field of information retrieval, a recommender system is any system that is able to find entities in a dataset that may be of interest to the user. In contrast to search engines, recommender systems do not base their results on a *query*, instead relying on user profiles.

In the general case, a recommender system applies to a dataset described by a data model containing entities (such as users and items) and relationships (such as ratings and social links). In the simplest recommender system, the data consists of one relationship type connecting one or two entity types. In more complex cases, the dataset contains multiple relationship types connecting any number of entity types.

The simple case of only one relationship type corresponds to several well-studied recommendation subproblems, such as link prediction, collaborative filtering, citation analysis, etc. In the case of multiple relationship types, hybrid recommenders are normally used. Hybrid recommenders are typically described as combining several simple recommenders, each corresponding to one relationship type. This approach however has two problems: First, by considering only subsets of the dataset, latent relations between entities from different relationship types may be missed. Second, for each dataset, a new hybrid recommender has to be developed.

To avoid these problems we propose the Universal Recommender, a recommendation engine with the following features.

- It applies to datasets with any number of entity and relationship types.
- It learns all its parameters without human intervention.

The first requirement ensures the recommender can be applied to future datasets whose structure is as-yet unknown. The second requirement is more critical: Since a recommender using complex datasets has a large set of parameters, a complex recommender runs the risk of either overfitting the data or having very low recommendation quality. The main challenges of the Universal Recommender are therefore a series of machine learning problems related to the complexity of semantic networks.

We begin the paper by reviewing typical recommendation datasets and tasks, and give known solutions to specific recommendation problems. We then describe the unified data model and the general approach of the Universal Recommder, and introduce the machine learning and optimization problems that it will have to solve. We finish by describing the case study of an IPTV recommender system.

2 Complex Semantic Networks

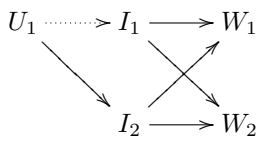
In this section, we give examples of classical recommendation scenarios motivating the comlpexity of typical recommendation datasets. We also review existing solutions for partial recommendation problems.

We describe all cases in the context of the Internet Protocol Television recommender system [17]. In the IPTV recommendation setting we want to recommend a TV program to a user. In the following examples we describe classical recommendation settings applied to our scenario.

2.1 Content-based Filtering

An early form of recommendation consists in content-based filtering. As in search engines, content-based filtering considers the content of items. While search engines require the user to enter a specific keyword for searching, content-based recommenders usually take keyword from another source, for instance a profile filled out by the user (containing words describing his interests), or from documents already seen or rated.

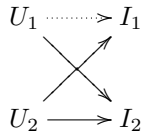
The following diagram shows a situation in which a user-item recommendation is found by analysis common features of two items. Arrows represent known relationships and the dotted arrow represents the relationship to predict. U, I and W represent users, items and words respectively.



For our IPTV system, each TV program would have a description of its content. The weight of single keywords for each TV program is then computed by the tf-idf measure. This example already shows a limitation of the content-based approach: Because the TV program descriptions are much shorter than typical documents, the tf-idf measure will be less accurate.

2.2 Collaborative Filtering

While the content-based approach is simple (being essentially a search engine), it requires the user to enter a profile. If our IPTV tracks the programs watched by each user, this information can be exploited directly giving collaborative filters.

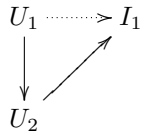


The general idea is the following: If we known which TV programs the user has seen, then we can use this information directly, without needing content information. To do this, a collaborative recommender system must consider the behavior of other users: If another user has seen the same TV program as a given user, we can recommend other TV programs seen by that other user. The result is a recommendation system that does not need any content information. Therefore, collaborative recommenders are often used in scenarios where no or little content is available: for instance movies or jokes [6]. For our IPTV recommender, this means we don't have to rely on content descriptions, and we may recommend TV programs that don't have a description.

What is more, a collaborative recommender can make use of explicit *ratings*. Compared to the typical "has-seen" information, ratings have the advantage of also admitting negative values, modeling dislike. To collect such ratings, we either ask users to explicitly rate TV programs, or derive ratings from user behavior. If for instance a user stops a TV program after just a few minutes, we may assign to it a negative rating.

2.3 Social Networks and Link Prediction

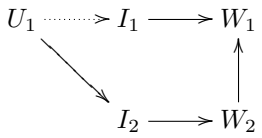
A further type of recommender is given when links are known between users. For instance, if IPTV users can maintain a buddy list, we may recommend the favorite items of a given user's buddies. This type of recommendation is particularly useful when trust is important. In this case, a trust measure can be defined between users, denoting the level of confidence a user has in another user's tastes.



In a addition to friendship and trust, which are positive relationships, we may allow users to mark other users as enemies, representing distrust [7, 15].

2.4 Lexicographic Information

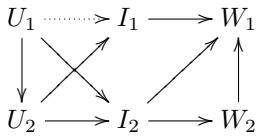
While words contained in descriptions may be used to find similar TV programs, the words themselves may be modeled as interlinked entities: Some words are synonyms, antonyms, etc. These relationships may be used to enhance a content-based recommender by also recommending documents of a related topic but using different glossaries.



We might even go so far as mapping words in different languages to each other, using information from a dictionary. This would allow the IPTV system to recommend programs in other languages about a known-liked topic.

2.5 Hybrid Recommenders

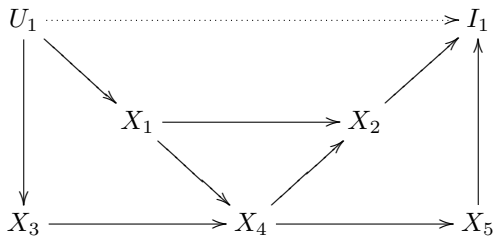
In many recommender systems, several of the previously described dataset types are known. For instance, a recommender system may have user-ratings for documents and at the same time content information about items. Recommenders that apply to such datasets are called *hybrid*. While hybrid recommenders exist for many combinations of entity and relationship types (e.g. [1, 2, 19, 24]), none of these can be applied to all complex network since they are not generic.



2.6 Semantic Networks

In the general case, datasets can be modelled as a set of entities connected by relationships. While in simple datasets are relationships are similar (e.g. the "has-seen" relation), complex networks almost always contain multiple relationship types. This is especially true when several datasets are combined.

For our IPTV recommender, we could for instance integrate information from DBpedia [3] or YAGO [22].



Semantic networks are the domain of the Universal Recommender: while hybrid recommenders apply to individual complex networks, the Universal Recommender is designed to apply to *any* complex network. This characteristic makes our approach *universal*.

3 Unified Representation of Datasets

In order to write a recommender system that applies to the use cases described in the previous section, we give a unified representation of datasets on which the Universal Recommender will be applied.

- Datasets consist of entities and relationships connecting two or more entities.
- Entities are grouped into multiple entity types.
- Relationships are grouped into multiple relationship types, each connecting a predefined number of fixed entity types.
- Relationships may be symmetric or asymmetric, corresponding to directed and undirected relations.
- Relationships may be weighted, and weights may be negative.
- Entities and relationships may both be annotated with attributes, for instance timestamps of ratings or the age of users.

Note that this definition includes not only binary relationships, but also higher-order relationships (e.g. tag assignments between users, tags and items.) Table 1 gives some examples of relationship types between users, items and words. Table 2 gives examples of relationship types by the number of different entity types they connect (unipartite, bipartite) and the range of edge weights. Table 3 gives an overview of traditional data mining application that can be interpreted as special cases of recommendation.

Table 1: Commons relationship types in recommender systems, arranged by the entity types they connect. Only unipartite and bipartite relationship types are shown. Unipartite relationship types are on the diagonal.

	User	**Document**	**Word**
User	Social network Trust network Email network Profile ratings		
Document	Explicit feedback Implicit feedback Authorship Commercial selling data	Citations Hyperlinks	
Word	Search history	tf-idf Categories	WordNet

Table 2: Common relationship types by the number of entity types they connect and the range of admitted edge weights.

	Number of connected entities		
	2		**3**
	Unipartite	**Bipartite**	**Tripartite**
Unweighted	Friendship	Authorship Categories	Folksonomy
Weighted	Profile rating Trust levels	Rating	
Positive	Communication	View history	Clickthrough data
Signed	Friend/foe network Trust/distrust	Like/dislike	Contextual rating

Table 3: Compilation of common relationship types, the entity types they connect, and the traditional data mining application using only that relationship type as a dataset.

Dataset	**Relationship types**	**Entity types**	**Weight range**	**Application**
Explicit feedback	Ratings	Users, items	$(1, \ldots, 5)$	Collaborative filtering
Implicit feedback	View, save, etc.	Users, items	Unweighted	Collaborative filtering, recommendation
Features	tf-idf	Text documents, phrases	Positive	Document classification
Social network	Friendship, enmity	Users	Unweighted or $\{-1, +1\}$	Link (sign) prediction, community building
Web	Hyperlinks	Web pages	Unweighted	Link prediction, ranking
Citation network	References	Scientific publications	Unweighted	Bibliographic analysis
Trust network	Trust, distrust	Users	Unweighted or $\{-1, +1\}$	Trust measures
Interaction network	Communication (e.g. email)	Users	Positive integer	Link prediction, network analysis
User profile ratings	Ratings of user profiles	Users	Signed	Expert finding, dating recommendation
Collaboration graph	Authorship	Authors, publications	Unweighted	Community building
Search history	Searches	Users, queries, phrases	Unweighted	Query reforming, query recommendation
Taxonomy	Category meembership	Items, categories	Unweighted	Classification
Lexical network	Synonymy, antonymy, etc.	Words		Query reforming, lexical search

4 General Approach: Latent Representation

In this section, we describe the general recommendation approach, which has many parameters. The general idea consist in representing entities in a latent space, in which relationships can be predicted by using the scalar product. In other words, if we associate a vector of length k to every entity, we can compute a prediction between two entities using the scalar product. This approach has two consequences:

- Computation of the latent model can be interpreted as a decomposition of the adjacency matrix of the complete network, allowing us to use known graph kernels.
- While computation of relationship prediction is fast, a method has to be found to map these predictions to recommendations. This is described in the next section.

The following recommendation algorithms (in the general sense) can be described as using a latent model:

- The singular value decomposition and eigenvalue decomposition [23], principal component analysis, latent semantic indexing.
- Various graph kernels [8, 9, 13, 18]. These can be applied to the eigenvalue or singular value decomposition of graphs, and can be learned separately from model building [14].
- Probabilistic approaches such as probabilistic semantic latent analysis [10–12], and latent Dirichlet allocation [4].
- Other matrix decompositions such as nonnegative matrix factorization [16], maximum margin matrix factorization [21], low-rank approximations with missing values [20].
- Methods based on the Laplacian matrix such as the commute time and resistance distance.
- Tensor decompositions such as the higher-order singular value decomposition, parallel factor analysis (PARAFAC), the Tucker decomposition.

4.1 Example

For the example of the IPTV recommender, we give a derivation using the singular value decomposition. Let U be the user set, I the item set and W the set of words. Then the dataset is given by the following adjacency matrices: the ratings $\mathcal{R} \in \mathbb{R}^{U \times I}$, the buddies $\mathcal{B} \in \{0,1\}^{U \times U}$ and the features $\mathcal{F} \in \{0,1\}^{I \times W}$. The weighted matrix $\mathcal{R}$ is then normalized to $\bar{\mathcal{R}}$ and aggregated with the other adjacency matrices into a single matrix $A \in \mathbb{R}^{(U+I+W)\times(U+I+W)}$.

$$A = \begin{pmatrix} w_B\mathcal{B} & w_R\bar{\mathcal{R}} & \\ w_R\bar{\mathcal{R}}^T & & w_F\mathcal{F} \\ & w_F\mathcal{F}^T & \end{pmatrix}$$

where $w_* > 0$ is the weighting of relationship type $*$.

This matrix is then decomposed, giving latent vectors for all three entity types. The approximation or decomposition used may be any of those described above. For simplicity, we adopt the notation of the singular value decomposition.

$$A = U\Sigma V^T = \begin{pmatrix} U_U \\ U_I \\ U_W \end{pmatrix} \Sigma \begin{pmatrix} V_U \\ V_I \\ V_W \end{pmatrix}^T$$

U_* and V_* are latent vectors of dimension $* \times k$, where k is the number of latent dimensions computed. These vectors can then be used for computing recommendations. To compute a rating prediction for the user-item pair (u, i), we would use $U_U(u) \cdot V_I(i)$.

5 Machine Learning Problems

Here we describe the machine learning problems associated with generic semantic recommender systems. While in unirelational networks a matrix decomposition approach is a common procedure, its application to complex networks gives rise to some additional issues:

- Weights and sparsity pattern of different relationship types may be completely different, in which case each relationship type has to be *normalized* separately.
- Since edge weights are usually not comparable, the question of finding the relative weights w_* arises.

5.1 Learning Normalizations

In recommender systems that apply to a unirelational dataset with edge weights such as ratings, a common first step consists in additive normalization. Given edge weights (a_{ij}), additive normalization computes new edge weights $b_{ij} = a_{ij} - \tilde{a}_{ij}$, where $(\tilde{a}_{ij})$ is a simple approximation to (a_{ij}), e.g. a row or column mean.

In most recommenders, this step is usually kept simple, such as subtracting the overall rating mean. In complex networks, each weighted relationship type may need separate normalization, and the overall normalization problem becomes non-trivial as the number of parameter increases with the number of relationship types.

5.2 Learning Relative Weights

In unirelational datasets, all edges have the same semantics, and an algebraic or probabilistic decomposition algorithm can use this fact to compute a low-rank model of the data. In complex networks however, such an algorithm would implicitely assume that edges have the same semantics, which in practice only works if the different relationship types have a similar weight range and degree distribution.

In order to apply these algorithms to complex networks, the different relationship types have to be weighted separately. The weights w_* depend on the characteristics of the subnetwork (e.g. the degree distribution), but also on overall considerations such as whether a particular relationship type is even useful for recommendation. Different weights must also be applied to different relationship types connecting the same entity types.

These weights can be hardcoded using domain-specific knowledge, for instance in the IPTV case by knowing that ratings contribute more to recommendations than the "has-seen" relationship type, or they can be learned automatically. We therefore propose the Universal Recommender to learn relative weights automatically, in order to avoid being dependent on domain-specific knownledge, and to validate domain-specific knowledge if present.

An example of different relationship types connecting the same entity types are various relationship types connecting users and items: "has-seen", "has-recorded", "has-bookmarked". While a human IPTV expert could set these relative weights by hand, learning the weights is a worthwile machine learning problem in itself.

6 Optimization Problems

In addition to the machine learning problems that make sure that recommendations actually correspond to user expectations, the following optimization problems are solved to ensure the scalability of the Universal Recommender.

- The computation of the latent recommendation model must be asynchronous. In other words, updates to the data model must be incorporated into the recommender model without recomputation of the whole recommender model. In practice, the recommender model is built iteratively, and the data model is read at each iteration, ensuring that changes are incorporated into the recommender immediately.

- While rating predictions can be computed in constant time for a given recommender model (using the scalar product), computations of recommendations are more complex. The underlying problem consists of finding a vector maximizing the scalar product with a given vector. This problem is similar but not identical to the metric nearest-neighbor problems. A common approach consists in clustering the set of nodes.

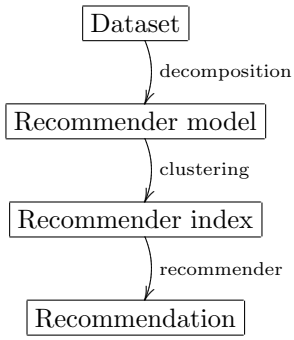


Figure 1: Computational flow diagram of the Universal Recommender. The dataset is first decomposed into a recommender model. In this model, entities are then clustered giving a recommender index. Finally, a recommender computes recommendations using the recommender index.

Diagram 1 shows the three-level recommendation procedure, in which the first two steps correspond to the optimization problems we describe in the next subsections.

6.1 Iterative Update of the Recommender Model

To compute recommendations in a dataset, a *recommender model* is built out of the dataset. This model building step may be slow, but the resulting model can be used to compute recommendations rapidly. If the dataset changes, for instance when users rate additional items, the model would have to be recomputed.

To avoid this overhead, we propose a recommender model that can be updated iteratively. In fact, most matrix decomposition or low-rank approximation problems can be solved iteratively, giving a recommender model where updates arise naturally from the update algorithm.

In this context, the role of the recommender model is analogous to PageRank for search engines [5]. The PageRank is a vector of entities (web pages) that can be updated by iterative algorithms (e.g. power iteration). In the case of the Universal Recommender, the model consists of a set of k vectors corresponding to the latent spaces of the rank reduction method. Updates can be performed in a way consistent with the underlying algorithm.

6.2 Recommendation Index

The goal of a recommender is to compute recommendations. Functionally, a recommender takes an entity as input (a user) and outputs a list of ranked entities (items). The recommender index however is meant to compute rating predictions. While rating prediction has received attention in itself (see e.g. the

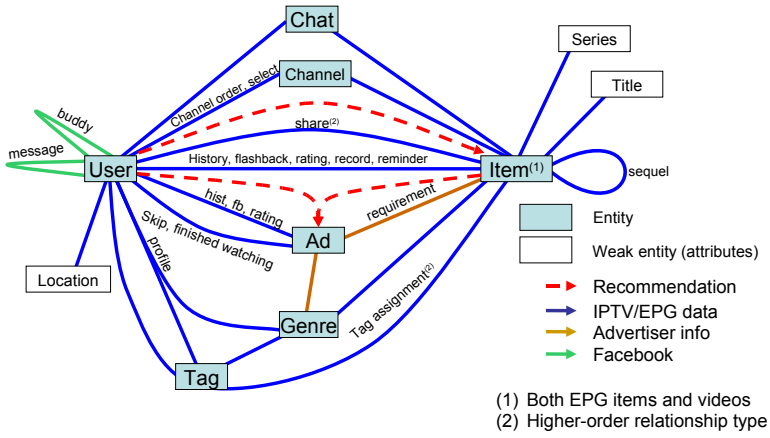


Figure 2: Entity-relationship diagram of the IPTV system. Entities are represented by nodes and relationships by connections. Recommendation use cases are drawn as dashed red lines. Any dataset usable by the Universal Recommender can be represented by such a graph.

Netflix Prize), they are useful to a recommendation system insofar as they can be used to rank items. To find to top k items that a user would rate with high scores, all n items have to be considered. Since runtime of recommendation should not depend on n, a *recommendation index* has to be used.

A recommendation index must solve the following problem: Given n vectors a_i and a vector x, find the top k such that $x \cdot a_i$ is maximal. A similar problem with the scalar problem replaced by the Euclidean distance is known as *nearest neighbor search*.

7 Case Study: IPTV

In this section we describe the IPTV recommender system as an example setting for the Universal Recommender.

In the Internet Protocol Television system, users can watch TV programs over the internet. In addition to the functionality provided by regular television, IPTV includes a semantic *recommender system* based on the Universal Recommender. Figure 2 shows the entities and relationships present in the IPTV system, along with the main recommendation scenario.

This example shows characteristics found in many recommender system

datasets: The primary entity types are users and items, which are TV programs in this case. The main relationship types connect users and items. In our example these are view, flashback, rating, record and reminder events. This scenario shows a common feature of recommender systems: several relationship connect the same entity types. Other relationship types connect secondary entities such as locations, genre, series and titles. User-user relationships are represented by message events and buddy lists, both common in recommender systems. This dataset also contains higher-order relationship types, in form of tag assignments and share events.

This example also shows how difficult it is in general to find build a good hybrid recommender system out of simple recommender system. There are so many relationship types that a hybrid recommender has too many parameters to be optimized by trial and error.

8 Conclusion

We proposed the Universal Recommender, a generic recommender system that applies to any dataset consisting of entity and relationships of any number of entity and relationship types. We showed how the Universal Recommender extends most known recommender systems using Internet Protocol Televion as a case study. We identified key machine learning and optimization problems that must be studied to implement a successful Universal Recommender.

As a real-world example illustrating use of the Universal Recommender, we presented the Internet Protocol Televion (IPTV) recommender.

References

[1] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Information Systems 23*, 1 (2005), 103–145.

[2] Basilico, J., and Hofmann, T. Unifying collaborative and content-based filtering. In *Proc. Int. Conf. on Machine Learning* (2004).

[3] Bizer, C., Cyganiak, R., Auer, S., and Kobilarov, G. DBpedia.org–querying Wikipedia like a database. In *Proc. Int. World Wide Web Conf.* (2007).

[4] Blei, D., Ng, A., Jordan, M., and Lafferty, J. Latent Dirichlet allocation. *Machine Learning Research 3* (2003), 993–1022.

[5] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[6] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval 4*, 2 (2001), 133–151.

[7] Guha, R., Kumar, R., Raghavan, P., and Tomkins, A. Propagation of trust and distrust. In *Proc. Int. World Wide Web Conf.* (2004), pp. 403–412.

[8] Gärtner, T., Le, Q. V., and Smola, A. J. A short tour of kernel methods for graphs. Tech. rep., 2006.

[9] Ham, J., Lee, D. D., Mika, S., and Schölkopf, B. A kernel view of the dimensionality reduction of manifolds. In *Proc. Int. Conf. on Machine Learning* (2004), p. 47.

[10] Hofmann, T. Probabilistic latent semantic analysis. In *Proc. Conf. on Uncertainty in Artificial Intelligence* (1999), pp. 289–296.

[11] Hofmann, T. Collaborative filtering via Gaussian probabilistic latent semantic analysis. In *Proc. Int. Conf. on Research and Development in Information Retrieval* (2003), pp. 259–266.

[12] Hofmann, T. Latent semantic models for collaborative filtering. *ACM Trans. Information Systems 22*, 1 (2004), 89–115.

[13] Kandola, J., Shawe-Taylor, J., and Cristianini, N. Learning semantic similarity. In *Advances in Neural Information Processing Systems* (2002), pp. 657–664.

[14] Kunegis, J., and Lommatzsch, A. Learning spectral graph transformations for link prediction. In *Proc. Int. Conf. on Machine Learning* (2009), pp. 561–568.

[15] Kunegis, J., Lommatzsch, A., and Bauckhage, C. The Slashdot Zoo: Mining a social network with negative edges. In *Proc. Int. World Wide Web Conf.* (2009), pp. 741–750.

[16] Lee, D. D., and Seung, S. H. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems* (2000), pp. 556–562.

[17] Members of the Open IPTV Forum. Open IPTV Forum, 2009. Whitepaper.

[18] Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E 74* (2006).

[19] Soboroff, I. M., and Nicholas, C. K. Combining content and collaboration in text filtering. In *Proc. Workshop on Machine Learning for Information Filtering* (1999), pp. 86–91.

[20] Srebro, N., and Jaakkola, T. Weighted low-rank approximations. In *Proc. Int. Conf. on Machine Learning* (2003), pp. 720–727.

[21] Srebro, N., Rennie, J. D. M., and Jaakola, T. S. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems* (2005), pp. 1329–1336.

[22] Suchanek, F., Kasneci, G., and Weikum, G. YAGO–a core of semantic knowledge. In *Proc. Int. World Wide Web Conf.* (2007), pp. 697–706.

[23] Zhang, S., Wang, W., Ford, J., Makedon, F., and Pearlman, J. Using singular value decomposition approximation for collaborative filtering. In *Proc. Int. Conf. on E-Commerce Technology* (2005), pp. 257–264.

[24] Zhu, S., Yu, K., Chi, Y., and Gong, Y. Combining content and link for classification using matrix factorization. In *Proc. Int. Conf. on Research and Development in Information Retrieval* (2007), pp. 487–494.