From Free-text User Reviews to Product Recommendation using Paragraph Vectors and Matrix Factorization

Georgios Alexandridis
School of Electrical and Computer Engineering
National Technical University of Athens
Zografou, Greece
gealexandri@islab.ntua.gr

Thanos Tagaris
School of Electrical and Computer Engineering
National Technical University of Athens
Zografou, Greece
thanos@islab.ntua.gr

Giorgos Siolas
School of Electrical and Computer Engineering
National Technical University of Athens
Zografou, Greece
gsiolas@islab.ntua.gr

Andreas Stafylopatis
School of Electrical and Computer Engineering
National Technical University of Athens
Zografou, Greece
andreas@cs.ntua.gr

ABSTRACT

Recent theoretical and practical advances have led to the emergence of review-based recommender systems, where user preference data is encoded in at least two dimensions; the traditional rating scores in a predefined discrete scale and the user-generated reviews in the form of free-text. The main contribution of this work is the presentation of a new technique of incorporating those reviews into collaborative filtering matrix factorization algorithms. The text of each review, of arbitrary length, is mapped to a continuous feature space of fixed length, using neural language models and more specifically, the Paragraph Vector model. Subsequently, the resulting feature vectors (the neural embeddings) are used in combination with the rating scores in a hybrid probabilistic matrix factorization algorithm, based on maximum a-posteriori estimation. The proposed methodology is then compared to three other similar approaches on six datasets in order to assess its performance. The obtained results demonstrate that the new formulation outperforms the other systems on a set of two metrics, thereby indicating the robustness of the idea.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Natural language processing**; **Maximum a posteriori modeling**; **Factorization methods**; **Learning latent representations**.

KEYWORDS

Recommender Systems; Collaborative Filtering; Matrix Factorization; Free-text reviews; Neural Language Processing; Word2Vec; Paragraph Vectors; Maximum A-posteriori Estimation

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316601

ACM Reference Format:
Georgios Alexandridis, Thanos Tagaris, Giorgos Siolas, and Andreas Stafylopatis. 2019. From Free-text User Reviews to Product Recommendation using Paragraph Vectors and Matrix Factorization. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3308560.3316601

1 INTRODUCTION

The evolution of Collaborative Filtering (CF) Recommender Systems (RS) in recent years has resulted in a paradigm shift, moving away from systems that are solely based on the ratings' matrix to systems that incorporate user generated free-text reviews in the recommendation process as well. This reality has been spearheaded by the widespread adoption of recommendation modules in various social media platforms, which allow their users to evaluate items (venues, products, etc.) not only using traditional rating scales (e.g like/dislike, 1 to 5 star ratings) but also providing textual reviews along with other meta-information (Figure 1). This augmented preference data may be further analyzed through the application of text analysis and opinion mining techniques in order to extract various aspects of the conveyed information.

Review-based CF RS may be categorized, with respect to how the available reviews are populated, into *review-based user profile building* systems and *review-based product profile building* systems [7]. The former construct a user profile out of the available free-text reviews (their operational principle being similar to user-based CF approaches), while the latter create descriptive item profiles out of the same corpus of user-generated reviews (their operation being analogous to item-based CF methodologies). The proposed framework in this work, on the other hand, attempts to merge the aforementioned approaches by constructing profiles for *both* the items and the users.

Textual reviews may be processed using various methods, such as the creation of term-based profiles, employing standard techniques like the *term frequency-inverse document frequency* (TF-IDF) formula. Other methodologies populate free-text reviews in an effort to extrapolate preference scores in predefined, discrete scales, usually through the adoption of sentiment analysis tools. Of course, textual reviews may be incorporated in existing recommendation

algorithms as an additional data dimension that extends the ratings' matrix. In this setting, the user opinion or the topic extracted out of the respective reviews may be used as an extra weighting scheme, when computing user and item similarities.

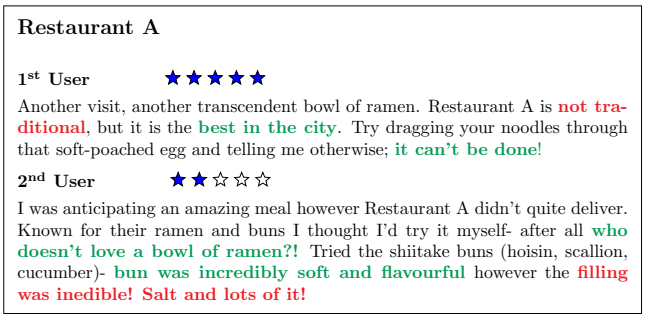


Figure 1: An example of free-text reviews of a restaurant

The majority of the aforementioned methodologies, however, place their emphasis on the word or the phrase level and many times fail to capture the whole context of the review. In Figure 1 for example, the phrase "not traditional" has a negative meaning, which is superseded by the next phrase ("best in the city"). The same review contains another phrase ("it can't be done!") that would have been labeled negatively by a sentiment analysis algorithm, but which, in fact, is used in a positive context. Weighting the appearances of positive and negative words and phrases is not very helpful either; e.g. the second review might contain much more positive than negative content but the last two phrases carry all the negative weight and justify the "poor" rating. Another interesting observation is that the phrase "Salt and lots of it!", which is used in an extremely negative context by the second reviewer, might fit a completely different narrative to another user who enjoys salty food.

Based on the reasoning above and extending earlier work [2], the approach outlined in this contribution is an efficient technique of enriching the user & item latent representations learned by CF-based ratings' matrix factorization algorithms, through the inclusion of the user-generated reviews into the procedure. This goal is achieved using a *neural language model* to map the free-text input of arbitrary size to a continuous real-valued numeric vector space. Unlike similar approaches that place the emphasis on the word or the phrase level, the novelty of the proposed system is that it learns the representations at the review level, using an appropriate model, the *Paragraph Vectors* (Section 3). In this way, it is possible to associate the same words and phrases, with a different meaning, to the various user and item profiles. This characteristic is exploited by *ParVecMF* (Section 5), a probabilistic matrix factorization algorithm applied to the combined information sources (ratings' matrix, vectorized representation of text) that learns the respective user & item latent vectors.

Additionally, Section 4 discusses matrix factorization methodologies, providing insight on the parameters and hyper-parameters involved and analyzing their influence on the produced user & item latent vectors. In Section 6, the strength of the proposed approach is assessed against three other techniques on a number of datasets and the obtained results (Section 7) on a set of two metrics and on three experiments ascertain the robustness of the overall framework. Finally, the paper concludes in Section 8, where possible future extensions are also discussed.

2 RELATED WORK

The incorporation of free-text user reviews into CF matrix factorization algorithms has been extensively studied in the relevant literature [7, 17]. However, most approaches attempt to extract the topic or the sentiment of textual reviews by examining the words they consist of, irrespective of the order they appear or their local context. In the most common scenario, each review is modeled as a distribution on a predefined number of latent ("hidden") topics and equivalently each topic is being defined by a distribution of words. This is the basis of the *Hidden Factors as Topics* (HFT) [14], the *Ratings Meet Reviews* (RMR), [12] and the *JMARS* [8] models.

Other approaches, like *TopicMF* [4], calculate the frequencies of words in user reviews and then construct a frequency matrix, which is factorized along with the ratings' matrix in a unified process. There are even cases where the *Paragraph Vector* model is also employed, like in the *Bag-of-Words regularized Latent Factor* model (BoWLF) [3]; yet the order of the words and their local context is not taken into account.

Some systems [1, 21] employ text parsers that perform a deep syntactic analysis of the reviews in order to discretize user sentiment with respect to various aspects of the items. These techniques do consider word order when extracting their context (e.g. positive, neutral, negative), but they do not encode the relation in-between words of the same context. Unlike neural language models, the aforementioned methodologies do not capture the writing *style* and therefore are not able to compute similarities and dissimilarities between free-text reviews in this dimension.

Neural language models, on the other hand, have been predominantly used in RS mainly as a means of achieving better representation of user preference in special cases (transactions [10], news stories [9], music tracks [22], check-ins at locations [18]).

Certain approaches, like *DeepCoNN++* [6] and *Collaborative Multi-Level Embedding* (CMLE) [25], do try to preserve both word order and word context. Nevertheless, they differ from *ParVecMF* in two key aspects; firstly, they employ pre-trained word embeddings on a general vocabulary (instead of training a specific NLP model for the given vocabulary) and secondly, they simply concatenate the embeddings of the words that make up the review, ignoring the *document* context. Finally, the *Language Model regularized Latent Factor* (LMLF) [3] model processes user reviews as a sequence of words, but it does not employ a neural language model for this task, using a recurrent neural network instead. To the best of our knowledge, our contribution is among the first to effectively account for word order & context, as well as document context at the same time, through the combination of paragraph vectors and CF matrix factorization in a unified learning approach.

3 THE PARAGRAPH VECTOR MODEL

The *Paragraph Vector* model [11] is an extension of the *Word2Vec* model [16] of distributed representations of words in a vector space.

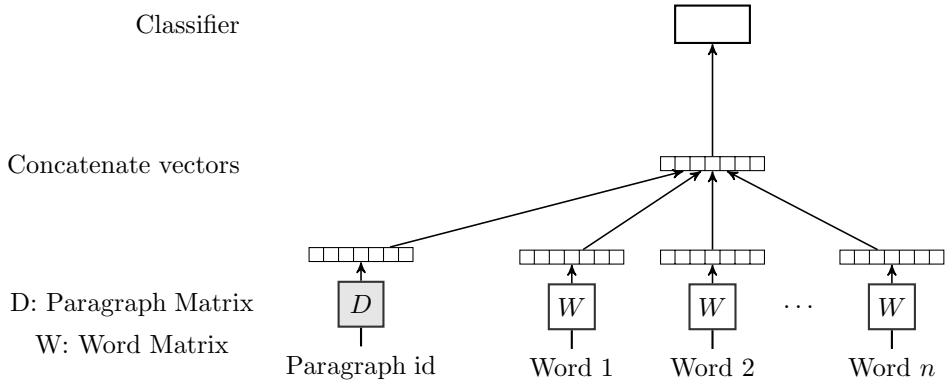


Figure 2: The Paragraph Vector model

In both models, every word is mapped to a unique vector, represented by a column in a word matrix W, indexed by the position of the word in the vocabulary.

Given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, the objective of both models is to maximize the average log probability of any given word in the sequence, conditioned on the appearance of the other words.

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_T|w_{t-k},\ldots,w_{t+k}) \tag{1}$$

Equation 1 above describes a prediction task, which is usually solved via the usage of a multiclass classifier such as *softmax*

$$p(w_T|w_{t-k},\ldots,w_{t+k}) = \frac{e^{y_{w_T}}}{\sum_i e_i^y}$$

Each term y_i is the un-normalized log-probability of each output word i, computed by

$$y = b + Uh(w_T|w_{t-k},\ldots,w_{t+k};W) \tag{2}$$

where U, b are the *softmax* parameters and function h is constructed by a concatenation (or averaging) of the word vectors extracted from W. For faster training, a *hierarchical softmax* with a *binary Huffman tree* structure is used in most cases [11]. Word vectors are usually trained via a neural network, using stochastic gradient descent and the gradient is obtained through backpropagation. After the training converges, the weights of each column of word matrix W correspond to the p dimensional vector representation of every vocabulary word.

The inspiration behind *Word2Vec* is that word vectors are asked to contribute to a prediction task about the next word in a sentence. In the *Paragraph Vector* model, paragraph vectors are used in a similar manner; they are asked to contribute to the prediction task of the next word given many contexts, sampled from the paragraph.

In addition to mapping each vocabulary word to a column in W, the *Paragraph Vector* model also maps every paragraph to a unique vector, represented by a column in a new document matrix D. The difference between *Paragraph Vectors* (Equation 2) and the *Word2Vec* model is that the output of the *softmax* classifier takes also into account the paragraph matrix D (Figure 2). Finally, the paragraph vector and word vectors are averaged (or concatenated) and are subsequently fed to the recommender system for the item prediction task.

A way to consider the paragraph token is as another word, which acts as a memory that remembers what is missing from the current context. This is equivalent to carrying the semantic information, or the topic, of the paragraph. For this reason, this model is called the *Distributed Memory* model of *Paragraph Vectors* (PV-DM). In this sense, the paragraph vectors, after the training phase, can be used simply as features for the paragraph.

Overall, the algorithm operates in two stages; the *unsupervised training* stage to get word vectors W and the *inference* stage to get paragraph vectors D. A third additional stage is to use D as a part of a probabilistic model for making predictions, like the one which is going to be presented in the subsequent section.

An important advantage of paragraph vectors is that they are learned from unlabeled data and thus can work well for tasks that do not have enough labeled data. They also address some of the key weaknesses of the bag-of-words models. Firstly, they inherit an important property of the *Word2Vec* model; the semantics of the words. In this space, we expect "orange" to be closer to "apple" than to "coffee". Secondly, they take into consideration the word order, at least in a small context, in the same way that an n-gram model with a large n would do.

Experimental results on two sentiment analysis tasks are presented in [11]. The first one is performed on the `Stanford Sentiment Treebank` dataset [19, 20], which has 11,855 sentences taken from the movie review site *Rotten Tomatoes*[1]. Every sentence in the dataset has a label which goes from very negative to very positive in the $[0-1]$ scale. The second one is performed on the `IMDB` dataset [13], consisting of 100,000 movie reviews with positive or negative labels. The authors report error rates that are by 16% and 15% better (relative improvement) than the best previous results on the two datasets, respectively.

The aforementioned advantages and the state-of-the-art results of the *Paragraph Vector* model have led us to choose this approach for the representation, in a vector space, of the textual information of our recommendation framework. However, it is used in a different context compared to the one described in [11]: rather than item descriptions, we analyze the items *reviews* made by the users. Then,

[1] https://www.rottentomatoes.com/

using this model, we try to extract the entire review's semantics with the ultimate goal of projecting similarly minded reviews in close distance in the vector space.

4 MATRIX FACTORIZATION

Matrix Factorization (MF) techniques approximate the mostly sparse and high dimensional ratings' matrix as the product of two other matrices, which are of lower dimensionality and more dense. More formally, given a positive semi-definite ratings' matrix $R \in \mathbb{R}_{+}^{n \times m}$ containing the ratings of n users on m items, the objective of the MF algorithm is to compute the elements of matrices $U \in \mathbb{R}^{n \times k}$ and $V^{\top} \in \mathbb{R}^{k \times m}$ so that their product $\widetilde{R}$ is as "similar" to R as possible, where similarity is defined in terms of a distance metric (e.g. the Euclidean distance).

$$R \approx \widetilde{R} \equiv UV^{\top} \tag{3}$$

The elements of matrix U (row vectors) may be regarded as the latent user factors that quantify the interests of each user. In a similar fashion, the elements of matrix $V^{\top}$ (column vectors) are regarded as the latent item factors that designate the opinion of the community about each item. It should also be noted that the length of the feature vectors (dimension k) is much smaller than both n and m ($k << n, m$).

In MF models, predictions are generated by multiplying the respective user and item latent vectors. That is, given a user i and an item j not rated by i, the predicted preference score $\widehat{r_{ij}}$ is equal to the inner product of the user latent vector $\mathbf{u}_i \in \mathbb{R}^{1 \times k}$ times the item latent vector $\mathbf{v}_j^{\top} \in \mathbb{R}^{k \times 1}$

$$\widehat{r_{ij}} = \mathbf{u}_i \mathbf{v}_j^{\top} \tag{4}$$

The most popular MF technique used in RS is the *Alternating-Least-Squares with Weighted-λ-Regularization* (ALS) algorithm [26]. In this approach, learning the latent representations of the user & item features is expressed as a minimization problem

$$\min_{\mathbf{u}_i, \mathbf{v}_j^{\top}} \left[\sum_{(i,j) \in R} (r_{ij} - \mathbf{u}_i \mathbf{v}_j^{\top})^2 + \lambda \left(||\mathbf{u}_i||^2 + ||\mathbf{v}_j||^2 \right) \right] \tag{5}$$

for all user and item pairs (i, j) in R (please note that a slightly different notation than in [26] is used).

The optimization problem described in Equation 5 above is non-convex since both $\mathbf{u}_i, \mathbf{v}_j^{\top}$ are unknown. However, if one of the unknowns is fixed to a constant value, then the same problem becomes quadratic and can be solved by a least-squares method. The ALS algorithm, at each iteration, interchangeably fixes one variable while optimizing the other, until convergence is achieved. In order to avoid overfitting, the magnitudes of the obtained solutions for $\mathbf{u}_i, \mathbf{v}_j^{\top}$ are penalized (regularized) by a certain λ term, which constitutes a hyper-parameter of the approach.

4.1 Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) is a special case of MF in which the underlying assumption is that the elements of matrices U, V stem from a probability distribution of a known type but of unknown parameters. The most common choices are the Normal and the Poisson distributions. In case of the former, the element r_{ij} of the ratings' matrix is thought to originate from the Normal distribution with a mean value of $\mathbf{u}_i \mathbf{v}_j^{\top}$ and a constant specificity of c (commonly set to $c = 1$ [26])

$$r_{i,j} \sim \mathcal{N}(\mathbf{u}_i \mathbf{v}_j^{\top}, c^{-1}) \tag{6}$$

Vectors $\mathbf{u}_i, \mathbf{v}_j$ constitute the parameters of a system described by the data r_{ij} and therefore can be approximated via the standard *Bayesian inference rule*

$$P(\mathbf{u}_i, \mathbf{v}_j | r_{ij}) \propto P(r_{ij} | \mathbf{u}_i, \mathbf{v}_j) \times P(\mathbf{u}_i) \times P(\mathbf{v}_j) \tag{7}$$

where $P(\mathbf{u}_i), P(\mathbf{v}_j)$ designate the *a-priori* distributions of the parameters. The system described in Equation 7 may be learned using *Maximum A-posteriori* (MAP) estimation, which tries to maximize the logarithm of the *posterior*

$$\begin{aligned} \arg\max_{\mathbf{u}_i, \mathbf{v}_j} \log P(\mathbf{u}_i, \mathbf{v}_j | r_{ij}) \quad &\propto \quad \arg\max_{\mathbf{u}_i, \mathbf{v}_j} \log P(r_{ij} | \mathbf{u}_i, \mathbf{v}_j) \\ &+ \quad \arg\max_{\mathbf{u}_i} \log P(\mathbf{u}_i) \\ &+ \quad \arg\max_{\mathbf{v}_j} \log P(\mathbf{v}_j) \end{aligned} \tag{8}$$

5 PARVECMF

In the proposed architecture, the main objective is to combine the *Paragraph Vector* model (Equation 1), which maximizes the log-likelihood of the appearance of any word in the text, conditioned on its surrounding words, and maximum a-posteriori estimation (Equation 8), which maximizes the log-likelihood of a specific user evaluating a particular item with the specified rating score, given the respective user & item latent vectors. Since the two objectives are different, a methodology should be devised in order to integrate them in a unified optimization scheme.

Initially, the free-text reviews are processed by the *Paragraph Vector* model, in two independent learning procedures; one that accumulates the reviews on a per user basis and another that accumulates them on a per item basis. In either case, the context of an individual textual review is determined by the respective rating score. The advantages of this approach are twofold; on the item level, the system amasses the opinion of the community for the item at-hand, along with higher-level characterizations that may have not been included in the item's description in the first place. On the user level, the system learns the "vocabulary" of each user (the words/phrases used when positive or negative feelings are expressed) along with each user's particular taste (recall the discussion on the salty food in Section 1).

Subsequently, and inspired by similar joint optimization approaches (e.g. [23], [25]), the elements of the user latent vectors are assumed to originate from the neural embeddings of the user's reviews ($\boldsymbol{\theta}_i \in \mathbb{R}^k$) plus and offset ($\boldsymbol{\epsilon}_i \in \mathbb{R}^k$) that aggregates the two different optimization objectives. In practice, this offset models the relationship between the user's rating and reviewing behavior

$$\begin{aligned} \mathbf{u}_i &= \boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i \\ \boldsymbol{\epsilon}_i &\sim \mathcal{N}(\mathbf{0}, \lambda_i^{-1} I_k) \end{aligned} \tag{9}$$

The offset ($\boldsymbol{\epsilon}_i$) is also assumed to originate from the multivariate normal distribution with a mean vector of $\mathbf{0} \in \mathbb{R}^k$ and a diagonal covariance matrix of $\lambda_i^{-1} I_k \in \mathbb{R}^{k \times k}$. The covariance matrix is chosen to be diagonal because, by definition, in MF the features are independent of one another.

In a similar manner, the latent vector for item j is derived from the neural embeddings of all the reviews it has received ($\boldsymbol{\theta}_j \in \mathbb{R}^k$), plus an offset ($\boldsymbol{\epsilon}_j \in \mathbb{R}^k$) that models the relationship between the ratings of the item and the said reviews

$$\begin{aligned} \mathbf{v}_j &= \boldsymbol{\theta}_j + \boldsymbol{\epsilon}_j \\ \boldsymbol{\epsilon}_j &\sim \mathcal{N}(\mathbf{0}, \lambda_j^{-1} I_k) \end{aligned} \tag{10}$$

Since the neural embeddings $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j$ described in Equations 9-10 above are learned by the *Paragraph Vector* model in a separate process, they are considered to be part of the data of the model (along with the rating value r_{ij}). Therefore, the Bayesian inference rule of Equation 7 becomes

$$P(\mathbf{u}_i, \mathbf{v}_j | r_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \propto P(r_{ij} | \mathbf{u}_i, \mathbf{v}_j) \times P(\mathbf{u}_i | \boldsymbol{\theta}_i) \times P(\mathbf{v}_j | \boldsymbol{\theta}_j) \tag{11}$$

Combining Equations 6 through 11 and re-writing the offsets $\boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_j$ as the difference between the user & item latent vectors $\mathbf{u}_i, \mathbf{v}_j$ and their respective neural embeddings $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j$, yields

$$P(\mathbf{u}_i, \mathbf{v}_j | r_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \propto \mathcal{N}(\mathbf{u}_i \mathbf{v}_j^\top, c) \times \mathcal{N}(\mathbf{0}, \lambda_i^{-1} I_k) \times \mathcal{N}(\mathbf{0}, \lambda_j^{-1} I_k) \tag{12}$$

and the log-likelihood $\mathcal{L}$ of the a-posteriori probability becomes

$$\begin{aligned} \mathcal{L} = &- \sum_{i=1}^{n} \frac{\lambda_i}{2} (\mathbf{u}_i - \boldsymbol{\theta}_i) I_k (\mathbf{u}_i - \boldsymbol{\theta}_i)^\top \\ &- \sum_{j=1}^{m} \frac{\lambda_j}{2} (\mathbf{v}_j - \boldsymbol{\theta}_j) I_k (\mathbf{v}_j - \boldsymbol{\theta}_j)^\top \\ &- \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{c}{2} (r_{ij} - \mathbf{u}_i \mathbf{v}_j^\top)^2 \end{aligned} \tag{13}$$

The MAP estimates for the optimal values of the elements of the user & item latent vectors are located at those points where the gradient of $\mathcal{L}$ with respect to the parameters of the model is equal to zero, leading to the following update rules

$$\mathbf{u}_i \leftarrow \left[c \mathbf{r}_i V + \lambda_{\mathbf{u}_i} \boldsymbol{\theta}_i\right] \left[c V^\top V + \lambda_{\mathbf{u}_i} I_k\right]^{-1} \tag{14}$$

$$\mathbf{v}_j \leftarrow \left[c \mathbf{r}_j U + \lambda_{\mathbf{v}_j} \boldsymbol{\theta}_j\right] \left[c U^\top U + \lambda_{\mathbf{v}_j} I_k\right]^{-1} \tag{15}$$

The graphical model of the system is illustrated in Figure 3.

An interesting observation on Equations 14-15 above is that the magnitude of regularization hyper-parameters λ_u and λ_v determine the extend to which the free-text reviews (in the form of the computed neural embeddings) contribute to the formation of the user & item latent vectors. Setting any (or both) of them to zero would result in the respective user & item latent vectors to reflect the information contained in the ratings' matrix only. On the other hand, if λ_u, λ_v assume very large values, then the effect of the ratings in the factorization process will be diminished and the learned latent vectors will be solely based on the representation of the free-text user reviews in the document vector space. Obviously, when $\lambda_u = \lambda_v = 1$ the ratings and the textual reviews have an equal impact on the optimization process.

6 EXPERIMENTS

6.1 Datasets

A set of experiments has been performed on number of datasets selected from two distinct collections. The first one is the `Amazon product data` collection of datasets [15], extracted from Amazon

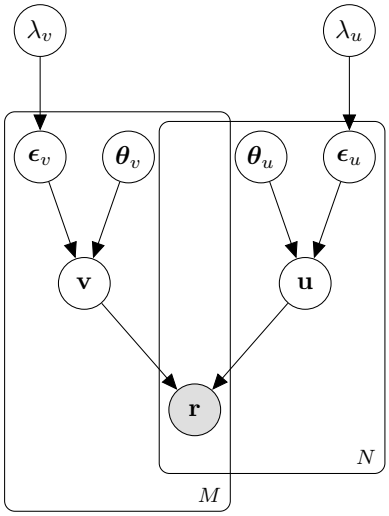


Figure 3: The graphical model of the proposed system

over a period spanning almost 20 years, while the second one is the `Yelp Open Dataset` [24], which contains more than 5,200,000 reviews on about 174,000 businesses from 11 metropolitan areas along with more than 200,000 pictures, provided by the respective online service.

Table 1: Dataset characteristics

Dataset	Users	Items	Reviews	M.w.+
A. `Amazon product data`				
1. `Movies and TV`	123,960	50,052	1,697,533	86
2. `Electronics`	192,403	63,001	1,689,188	64
3. `CDs and Vinyl`	75,258	64,443	1,097,592	119
4. `Kindle Store`	68,223	61,934	982,619	62
5. `Home and Kitchen`	66,519	28,237	551,682	60
B. `Yelp Open Dataset`				
1. `Yelp Reviews`*	213,170	94,303	3,277,932	88

+ Median words per review, * reduced to 5-core

Table 1 outlines the characteristics of the datasets used in the experiments, while Table 2 summarizes the available fields of each review entry in every dataset. Five of the largest datasets in the `Amazon product data` collection were chosen, along with the reviews subset of the `Yelp Open Dataset`. It should be noted that the Amazon datasets are in the form of 5-core; that is, ever user/item in the datasets has contributed/received at least 5 reviews. In order for the results to be comparable, the Yelp dataset has also been reduced to 5-core.

As it is evident from the aforementioned tables, each review is very descriptive in terms of the available information and meta-information. Nevertheless, the datasets remain extremely sparse, having all of them a data density of $0,01\% - 0,03\%$ (computed using the standard ratio of the number of reviews over the product of the number of users times the number of items) and the distribution

Table 2: Available fields per review

Field		Description
Amazon	Yelp	
asin	business_id	The unique item id
reviewerID	user_id	The unique user id
reviewerName	name	The user's name
helpful	useful	The ratio of users who found this review useful (Amazon) & the Number of useful votes (Yelp)
overall	stars	The rating of the product (5-star scale)
reviewTime	date	The time of the review
summary	-	Short summary of the review, written by the reviewer himself
reviewText	text	The actual text of the review
-	funny	Number of funny votes this review received
-	cool	Number of cool votes this review received

of the number of reviews over the number of users and items has *long-tail* characteristics [5].

6.2 Procedure

The sparsity of the data and the other properties of the datasets are reflected on certain experimental design choices. As the both of them are eventually in the form of 5-core, the recommendation list size is fixed to 5 items. The relatively small number of median words per review (5^{th} column of Table 1) also means that all words are to be considered by the *Paragraph Vector* model, therefore the minimum word frequency is set to one. The remaining parameters of the *Paragraph Vector* model (window size and number of epochs) are set to some widely-adopted predefined values (5 and 10 respectively).

In order to assess the effectiveness of *ParVecMF*[2], it is compared to three other systems; *ALS* [26] (Section 4), a pure matrix factorization approach, *HFT*[3] [14] (Section 2), which incorporates the free-text reviews in MF, modeling each word irrespective of the others as a distribution of topics and finally *DeepCoNN++*[4] [25] (Section 2), which preserves word order and context but does not take into account the document (review) context. The hyper-parameters of the other three approaches have been set to the optimal values reported by their respective authors, with the exception of the pre-trained vectors used in *DeepCoNN++*, which were chosen to be of the same dimensionality as the paragraph vectors in order for the comparisons to be fair.

6.3 Metrics

The performance of all systems is measured on two rank metrics, an information-retrieval one (*Mean Average Precision at N* or MAP@N) and an accuracy one (*Mean Reciprocal Rank at N* or MRR@N). Both metrics are defined over the total number of recommended lists of size N for the users in the test set T. Mean average precision is the mean of average precision for each recommended list

$$MAP = \frac{1}{|T|}\sum_{u=1}^{|T|}\overline{Pr(u)}$$

where the average precision in a list of length L for a user u is defined as

$$\overline{Pr(u)} = \frac{1}{|I_u|}\sum_{i=1}^{L} Pr(i) \times rel(i)$$

with I_u being the set of relevant items for user u, $Pr(i)$ the precision at cut-off i in the list and $rel(i)$ is equal to one if the i-th item in the list is relevant for u (and zero otherwise).

The mean reciprocal rank is defined as

$$MRR = \frac{1}{|T|}\sum_{l=1}^{|T|}\frac{1}{\text{rank}_l}$$

where rank_l designates the position of the first relevant item for the l-th user in the recommendation list. For both metrics and for the datasets at hand, the relevance threshold is set to four out of five stars. It should also be noted that MAP and MRR are computed for the "good" items in the test set (those items in the test set above the relevance threshold). Finally, the evaluation methodology followed is 5-fold cross validation computed over the available evaluations.

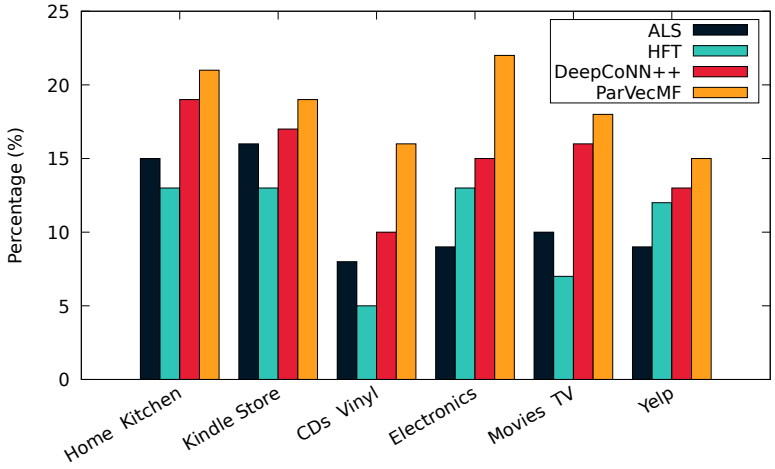
7 RESULTS

Figures 4-6 summarize all experimental results. All metrics are averaged over the 5 folds, with the statistical significance of all results being asserted by the *Wilcoxon singed rank test* ($p_{value} < 0.01$), for all possible model pairs on all datasets. More specifically, Figure 4 displays the results of the evaluation procedure for latent user/item vectors of 50 features and for a recommendation list of 5 items.

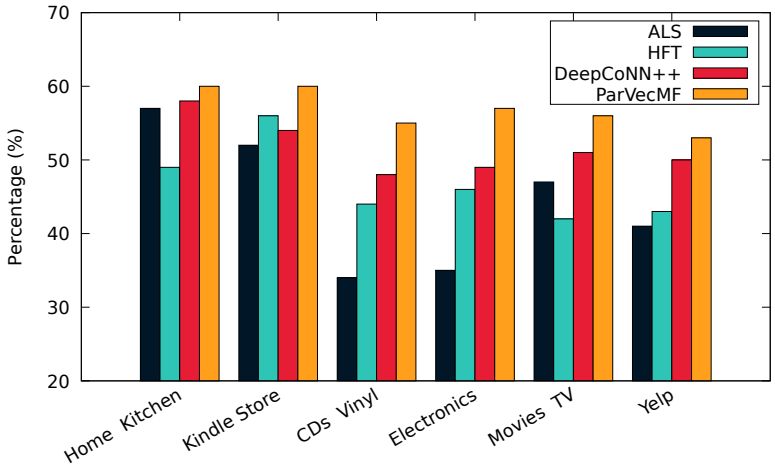
Figure 4a exhibits the results of the MAP metric. As it is evident, *ParVecMF* outperforms the other approaches on all datasets, proving the robustness of the methodology. An interesting observation is that the pure MF algorithm (*ALS*) performs better than the simpler approach (*HFT*) on most datasets, an indication that words should not be processed irrespective of their local context. Additionally, *DeepCoNN++* is surpassed by our system on all datasets. This behavior is attributed to two factors; firstly, *DeepCoNN++* takes into consideration only the word context and it misses the review context, as it merges all reviews of each user/item in its input, regardless of their actual polarity. Secondly, it is based on pre-trained vectors of a general vocabulary instead of training an NLP model on the specific vocabulary used in each dataset.

Figure 4b summarizes the performance of all examined systems on the MRR metric. Again, the proposed system exhibits a clear performance lead, which means that it is, on average, more capable of recommending at least one interesting item to the users. With the exception of the `Home & Kitchen` dataset, this conclusion also holds true for the other two systems that process free-text reviews (*HFT*, *DeepCoNN++*) when compared to the pure MF approach (*ALS*).

[2]https://git.islab.ntua.gr/gealexandri/parvecmf
[3]http://cseweb.ucsd.edu/ jmcauley/code/code_RecSys13.tar.gz
[4]https://github.com/chenchongthu/DeepCoNN



(a) Mean Average Precision for a list of 5 items (MAP@5)



(b) Mean Reciprocal Rank for a list of 5 items (MRR@5)

Figure 4: Results on all datasets (50 features)

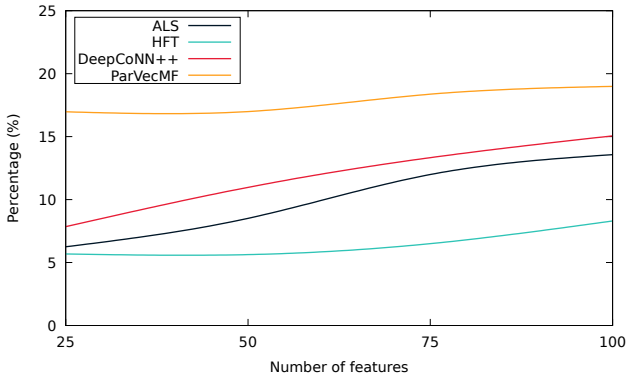
Therefore, including textual reviews as an extra input dimension generally boosts RS performance.

In another experiment, a dataset of medium size has been selected (`Cds and Vinyl`) and the algorithms are evaluated on the chosen set of metrics for a varying length of the latent user & item vector space. Figure 5a illustrates the increase of the MAP metric for all systems examined, as the length of the latent user & item vectors grows. The performance of *ALS* and *DeepCoNN++* is more dependant on the size of the feature space, while the proposed approach is robust even at smaller feature spaces. *HFT* efficiency, on the other hand, remains low at small feature spaces and starts to gradually increase only for larger spaces. In Figure 5b, the respective change of the MRR metric with respect to the feature space size is demonstrated. It is obvious that the ability of all algorithms to produce at least one meaningful recommendation is greatly affected by the length of the feature space and the larger it becomes, the more informative the obtained user/item latent features are.

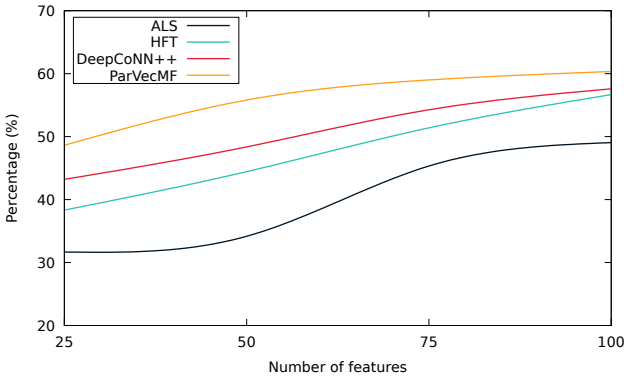
Finally, in Figure 6, the effect of the regularization parameters λ_u and λ_v, which determine the extend to which the free-text reviews contribute to the user & item latent vectors, is assessed on *ParVecMF* for the `Home and Kitchen` dataset. Each time, one of λ_u, λ_v is fixed to 1 and the value of the other is alternated between very small (0.01) and large (10) values. When λ_u, λ_v are individually set to a very small value, it means that the respective user/item neural embeddings play a minimal role in the optimization process of Section 5 and when they are set to a large value, then they dominate the said process. As expected, the performance of both metrics deteriorates in the latter case, meaning that no specific component of the proposed framework should overshadow the other.

8 CONCLUSION

This work outlines a new methodology of combining user reviews, in the form of neural embeddings, and ratings in probabilistic matrix

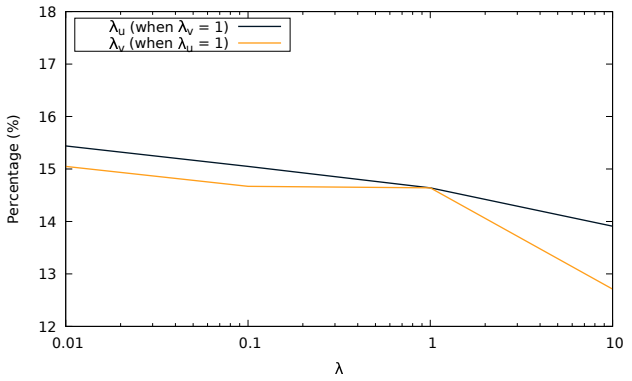


(a) Mean Average Precision for a list of 5 items (MAP@5)

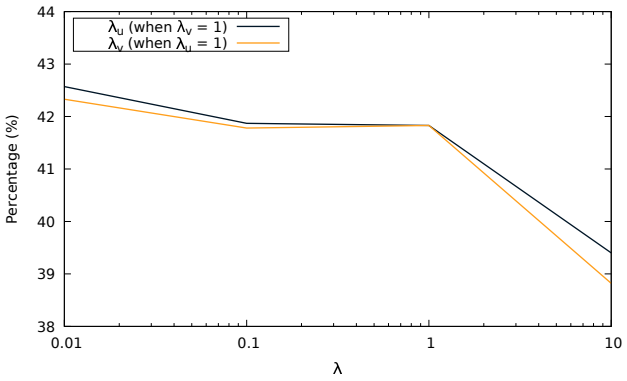


(b) Mean Reciprocal Rank for a list of 5 items (MRR@5)

Figure 5: The effect of the feature space on all systems (`CDs and Vinyl` dataset)



(a) Mean Average Precision for a list of 5 items (MAP@5)



(b) Mean Reciprocal Rank for a list of 5 items (MRR@5)

Figure 6: The effect of the regularization parameters on *ParVecMF* (`Home and Kitchen` dataset)

factorization. The presented results on a variety of datasets support the robustness of the proposed method and clearly indicate that this is a research direction worth of further exploring. The main novelty of our contribution is the use of the *Paragraph Vector* model as a means of representing textual reviews given by the users themselves instead of encoding item descriptions or general user preference data, like other approaches do.

The usefulness of this design choice for recommender systems should be once again stressed, as similarities in the way items are reviewed by the users imply an affinity in taste. The *Paragraph Vector* model further stimulates this notion of likeness, as it expands the similarity in the context of reviews that consist of different words. This domain is ignored by representations based solely on the word level, either dependent on (*DeepCoNN++*) or independent of (*HFT*) word order. Additionally, the importance of training an NLP model on each specific domain in order to extract the respective semantics, in contrast to using pre-trained vectors on general vocabularies (*DeepCoNN++*), should also be highlighted.

In general, review-based recommender systems are expected to become more and more popular, as the amount and dimensions of the available information evolves with time. This fact means that neural language models are likely to find even more applications in recommender systems, especially in conjunction with domains such as the social networks.

Finally, the effectiveness of the presented model should be further assessed by performing more comparative experiments to other related approaches. It would also be interesting to extend the proposed framework in order to incorporate more information and meta-information (e.g.the helpfulness of each review), where available (Table 2).

REFERENCES

[1] S. Aciar, D. Zhang, S. Simoff, and J. Debenham. 2007. Informed Recommender: Basing Recommendations on Consumer Product Reviews. *IEEE Intelligent Systems* 22, 3 (May 2007), 39–47. https://doi.org/10.1109/MIS.2007.55

[2] Georgios Alexandridis, Georgios Siolas, and Andreas Stafylopatis. 2017. ParVecMF: A Paragraph Vector-based Matrix Factorization Recommender System. *CoRR* abs/1706.07513 (2017). arXiv:1706.07513 http://arxiv.org/abs/1706.07513

[3] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning Distributed Representations from Reviews for Collaborative Filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 147–154. https://doi.org/10.1145/2792838.2800192

[4] Yang Bao, Hui Fang, and Jie Zhang. 2014. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. AAAI Press, 2–8.

[5] Òscar Celma. 2010. *The Long Tail in Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 87–107. https://doi.org/10.1007/978-3-642-13287-2_4

[6] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 1583–1592.

[7] 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

[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] Nemanja Djuric, Hao Wu, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2015. Hierarchical Neural Language Models for Joint Representation of Streaming Documents and Their Content. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 248–255. https://doi.org/10.1145/2736277.2741643

[10] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1809–1818. https://doi.org/10.1145/2783258.2788627

[11] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Eric P. Xing and Tony Jebara (Eds.), Vol. 32. PMLR, Bejing, China, 1188–1196. http://proceedings.mlr.press/v32/le14.html

[12] Guang Ling, Michael R. Lyu, and Irwin King. 2014. Ratings Meet Reviews, a Combined Approach to Recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 105–112. https://doi.org/10.1145/2645710.2645728

[13] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 142–150. http://dl.acm.org/citation.cfm?id=2002472.2002491

[14] 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

[15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/2766462.2767755

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781

[17] Michael P. O'Mahony and Barry Smyth. 2018. *From Opinions to Recommendations*. Springer International Publishing, Cham, 480–509. https://doi.org/10.1007/978-3-319-90092-6_13

[18] Makbule Gulcin Ozsoy. 2016. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356* (2016).

[19] Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 115–124. https://doi.org/10.3115/1219840.1219855

[20] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Vol. 1631. 1642.

[21] Anna Stavrianou and Caroline Brun. 2015. Expert Recommendations Based on Opinion Mining of User-Generated Product Reviews. *Computational Intelligence* 31, 1 (2015), 165–183. https://doi.org/10.1111/coin.12021

[22] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 225–232. https://doi.org/10.1145/2959100.2959160

[23] Chong Wang and David M. Blei. 2011. Collaborative Topic Modeling for Recommending Scientific Articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, NY, USA, 448–456. https://doi.org/10.1145/2020408.2020480

[24] Yelp. 2018. The Yelp Open Dataset. https://www.yelp.com/dataset/

[25] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative Multi-level Embedding Learning from Reviews for Rating Prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 2986–2992. http://dl.acm.org/citation.cfm?id=3060832.3061039

[26] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM '08)*. Springer-Verlag, Berlin, Heidelberg, 337–348. https://doi.org/10.1007/978-3-540-68880-8_32