Project Number: 766186
Project Acronym: ECOLE
Project title: Experienced-based Computation: Learning to Optimise

Deliverable D3.3
Interpretable Models for Product Feature Optimization with User-generated Data

Authors:
Giuseppe Serra, Zhao Xu – NEC Laboratories Europe GmbH
Peter Tino, Xin Yao – University of Birmingham

Project Coordinator: Professor Xin Yao, University of Birmingham
Beneficiaries: Universiteit Leiden, Honda Research Institute Europe, NEC Laboratories Europe

H2020 MSCA-ITN
Date of the report: 31.03.2021

Contents

Acknowledgement

This deliverable was initially named "Text mining models for product feature optimization". The main goal was on developing statistical machine learning and text mining methods to automatically detect customer opinions on aspects and features of products starting from large collections of unstructured texts. However, user-generated data do not contain only texts, and considering this type of data solely would have been limiting for our goal. Furthermore, interpretability has become an increasingly important aspect in real-world applications recently. To make the topic of the deliverable more exhaustive, we included these two aspects in our work and changed the deliverable title into "Interpretable models for product feature optimization with user-generated data".

Executive summary

This document provides a concise report on the research invested and the scientific contributions made regarding the work package 3.3 in ECOLE. The objective of WP3.3 is to investigate methods for automatically detecting customer opinions on aspects and features of products starting from large collections of unstructured data, e.g. textual data. In ECOLE, the idea of learning human understandable representations with the help of supplementary user-generated information was proposed. The findings suggest that using the additional information contained in the unstructured user-generated data can help unveiling the user opinions and the product features while improving the interpretability of the associated latent representations.

Major Achievements

Major scientific achievements regarding the work package 3.3 are presented. In particular, short answers to some of the most important research questions are described:

Research Questions	Discussion
Can additional human-understandable data (e.g. textual data) be integrated for learning high-dimensional representations of items in order to improve their interpretability?	Our findings indicate that is possible to learn human-understandable vector representations from textual data (cf. Figure 3 and Figure 5).
How does learning interpretable vector representations affect the predictive performance of the models?	In our findings, learning interpretable vector representations does not affect the predictive performances. The results show that we are able to learn human-understandable vector representations while maintaining competitive predictive performances (cf. Table 2).

What are potential ways of relating the interpretable latent representations to application scenarios?	The presented approaches can be integrated with different learning tasks, as long as some textual information is available. Additionally, we believe that the learned latent representations can be directly used as model input for downstream tasks. Potentially, we could utilize user preferences encoded in the latent representations for other multi-criteria optimization frameworks.

1 Introduction

In the Experience-based Computation: Learning to Optimise (ECOLE) project, we aim to address several challenges in the automotive setting, including Optimisation using Natural Computation, Multi-objective optimisation and System Engineering and Big Data Analytics. The project has been further divided into several subprojects, where each early stage researcher (ESR) is responsible for each subproject. The research aims of ECOLE include shortening the product cycle, reducing the resource consumption during the complete process, and creating more balanced and innovative products. Instead of just developing technologies to solve a given problem, it takes a bold step forward and propose to use knowledge automatically across different problem domains. Referring to knowledge, skill, and practice derived from problem solving processes in time, the goal is to automatically learn and transfer the experience of optimizing one product or process for solving other optimization problems.

In Work Package 3 (WP3) of the ECOLE project, we focus our research on developing statistical machine learning and text mining methods for automatically detecting customer opinions on aspects and features of products, and for analyzing user behaviors starting from large collections of unstructured data. During the report period, the idea of learning human understandable representation with the help of supplementary user-generated information was proposed. Since user-generated data can be of different forms (e.g. texts, time series), we aim at developing different approaches considering the different nature of the data**.** In the context of ECOLE, the final target is to incorporate the valuable information contained in the user-generated into industrial applications.

General overview and motivations
User-generated data (e.g. social media actions, reviews in online retailers) contain a wealth of information. In the context of Industry 4.0, leveraging the rich latent information contained in the available user-generated data can be crucial for many purposes. Nowadays, research in the manufacturing industry focuses on developing advanced data mining approaches to discover hidden patterns, to predict market trends and to learn customer preferences and unknown relations, for improving their competitiveness and productivity. Thanks to scalable, high-performance infrastructures we usually rely on sophisticated machine learning models with the risk of creating and using decision systems that we do not really understand. Indeed, these techniques are essentially black boxes and do not have a strong theoretical foundation. Consequently, the results are poorly explainable. The latter is probably the best-known lack in this area and, depending on the domain, this problem can be crucial. For example, in a medical application, how can we trust a diagnosis suggested by an algorithm with high accuracy but without a complete understanding of the output? In automotive design, how can a manufactory invest on a new car design if we cannot investigate the reasons behind a given algorithmic decision? In business scenarios, how can a company make some possible profitable decision without understanding if the proposed move is the right one or not?

For this reason, considering the wide use of machine learning methods in real-world applications and the increased attention regarding the interpretability capacity of the models, we will also focus on developing methods to be interpretable, to make the resulting output more trustable.

The need for interpretable models

Recently, deep learning approaches are reaching impressive performances on a variety of different tasks (e.g. computer vision, text classification) **[1]**. Despite the good performances, there are still gray areas regarding these methods. Many works **[2] [3] [4] [5]** have proved how easy is to fool a deep learning method by altering an image or a text with some perturbations. For example, experiments show that some changes in an image would lead the model to classify it as something different (e.g. a tomato as a dog) or that is possible to produce images that the model would classify with high confidence but are completely not recognizable by humans (e.g. classifying a white noise as a tomato). The same has been proved for text analysis, where minimal perturbations in a text lead the model to not recognize the right topic. These results show some of the limitations of these models and raise reasonable concerns about their trustiness. This limitation is particularly serious when dealing with real-world applications and decisions that could have an economic or social impact. The wide use of machine learning methods in industrial, medical and socio-economical applications is requiring the research community to provide explanations about the results. Thus, interpretable AI has been increasingly receiving more and more attention across different scientific disciplines and industry sectors.

The need for integrating user-generated data in the engineering-data driven analytics for industry verticals

In the context of Industry 4.0, the user-generated data available online could have a primary role to discover market trends, user preferences and to improve market strategies and productivity. The final target of this work will be then to incorporate the valuable information contained in the user-generated data into a multi-domain framework for industrial applications (e.g. automotive process). Ideally, we could use this information to provide useful insights for product feature optimization. For example, in a product design problem, we could utilize user design preferences extracted from textual data for adding some aesthetic constraint to the multi-criteria optimization framework, in order to create a new design that could potentially satisfy users more than before while improving its aerodynamics.

For the limitations of the machine learning methods explained above, the results should be human-interpretable. To support humans in a cooperative process with AI, we need to develop models and techniques that are human-understandable, in order to assist them in the evaluation and decision steps. The interpretability of the results will help the experts to make decisions in an aware manner. In this way, the evaluation process of AI-based applications will be faster, easier and more reliable.

Summarizing, through our work, we have the following targets:

1. Detect user preferences and analyze user behaviors.
2. Exploit different types of user-generated data: e.g., texts and time series.
3. Investigate and develop interpretable models and outputs.
4. Integrate our findings in real-world applications and, in particular, industrial applications.

In this report, we present the current state of our research on interpretable models for user preference learning with user-generated data gathered from real-world scenarios.
In section 2, we describe the methodology for interpreting node vector representations using text-labeled graphs. First, we concisely report the state-of-the-art related to our work. Second, we will present our approach to improve the interpretability of node vector representations (usually not interpretable from the human perspective) using the additional textual information that, in many cases, is available along with the structure of the data (i.e. graph).
In section 3, we introduce a probabilistic framework for generating a model-based visualization tool to organize user and product latent classes with review-text information. Many works focused on designing accurate and fast algorithms for recommendation, but less attention on implementing systems for understanding and for visualization of user preference and product characteristic patterns using review data.
In both sections, the running example will be an e-commerce system, but the proposed approaches can be integrated with alternative scenarios having textual information.

In section 4, we discuss future ideas for learning user profiles starting from different types of user-generated data than textual, namely event point processes and multivariate time series. The ideas presented in this section are in progress, and we are planning to investigate them in the remaining period. Section 5 concludes the report.

2 Interpretable user profiling with text-labeled graphs.

Many real-world user data are represented in forms of graphs, e.g. social networks, recommender systems and citation networks, since the graphs can capture relations and interactions of the involved entities. In this scenario, the performances are usually improved by using methodologies that represent the graph entities through vector representations (i.e. embedding) while preserving structural information. These methods often map nodes into latent spaces and learn vector representations of the nodes for a variety of downstream tasks. To gain trust and to promote collaboration between AIs and humans, it would be better if those representations were interpretable for humans. For an unsupervised learning setting as node embedding, interpretation can be more complicated since the embedding vectors are usually not understandable for humans (Figure 1).

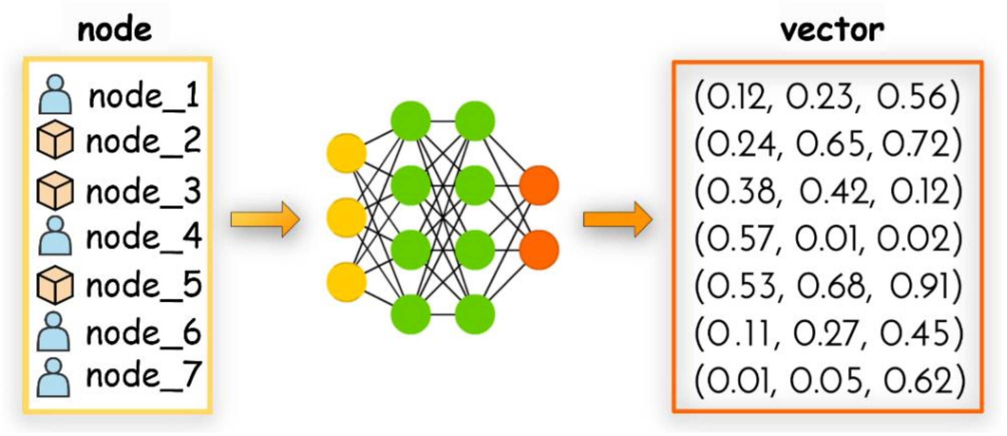


Figure 1 - Example of node embedding

On the other hand, in many cases, nodes and edges in a graph are often associated with textual data, e.g. reviews in user-product graphs, social media posts in social media or medical narratives in doctor-patient graphs. A question naturally arises: could we integrate the textual information in the learning phase to improve the interpretability of node embedding?

2.1 State of the art

The adoption of complicated machine learning methods (e.g.~deep neural networks) in real-world applications have recently led to an increasing interest in interpretable AI **[6]**. The extensive use of machine learning methods in industrial, medical and socio-economical applications requires a better understanding of these models. The improper use of machine learning could lead to high-impact risks since humans could not fully understand the underlying system or the resulting output. Recent advances in interpretable AI propose many post-hoc interpretability approaches and, depending on the setting and the application domain, they can be categorized as follows:

- *Global/local explanations*: LIME **[7]** and SHAP **[8]** are the most popular approaches in this category. They are surrogate models able to explain the outputs of any classifier in an interpretable way. For example, in a medical environment, they can visualize what are the symptoms in the patient's history that led to the prediction *'The patient has the flu'*.

- *Logic-based approaches*: the logic-based methods create falling rules and interpretable decision sets. In **[9]**, inspired by healthcare applications, the authors propose a Bayesian framework for learning falling rule lists that consist of an ordered list of if-then rules to determine which sample should be classified by each rule. **[10]** propose to learn decision sets (i.e. independent sets of if-then rules) through an objective function that simultaneously optimizes accuracy and interpretability of the rule.

- *Attribution methods*: in this broader category, the methods try to explain the output by highlighting characteristics of the output itself (e.g. textual explanations with highlighted relevant words) or by highlighting characteristics of the input that strongly influence the result. In the latter case, the most popular approaches attempt to analyze the gradient to investigate how the input features condition the output. **[11]**, **[12]** generate *saliency maps* in convolutional neural networks (CNNs). **[13]** estimates the influence of training examples in neural matrix factorization models.

In this work, we focus our attention on explaining outputs in a different unsupervised learning case, namely node embedding. There are few previous works attempting to improve the interpretability of node embeddings. The existing works mainly aim to explain the embedding dimensions as clusters in an implicit manner, e.g. employing Canonical Polyadic decomposition **[14]**, and assigning a meaning to each vector dimension **[15]**. Unlike these approaches, our method focuses on improving the interpretability of node embeddings explicitly to get human understandable explanation by exploiting the extra textual information associated with the graphs. Our method maps the latent space of node embeddings into textual space through word-based vectors. Thus, the additional available textual information works as a human-understandable source to generate explanations of node embeddings.

We present an approach to represent the latent space of node embedding into a textual space by exploiting the available human-understandable information (i.e. the textual information) to interpret the embedding vectors. Starting from a text-labeled graph, we integrate the textual information into the model to learn interpretable node embeddings. For each node $\boldsymbol{i}$, we generate a textual explanation that is formulated as a node-specific word distribution conditioned on its embedding vector $\boldsymbol{x_i}$. Since the creation of the word-vectors is directly linked with the node embeddings (which are supposed to work well in downstream tasks), we use an objective function that combines the accuracy of a downstream task (e.g. rating prediction, sentiment analysis) with the likelihood of the textual corpus. We introduce an additional node clustering to model the patterns among the embedding vectors, the corresponding textual explanations, and the associated texts. The additional cluster assignment allows our model to learn the discrete structure of the graph data. In summary, our model attempts to combine two objectives: a) learn node embeddings that perform well in downstream tasks b) generate textual explanations of the learned vector representations. To illustrate the method, we use a typical review network as a running example. Assume there is a bipartite graph $\boldsymbol{\mathcal{G}}$ with $\boldsymbol{N}$ number of users and $\boldsymbol{M}$ number of products. Between a user $\boldsymbol{i}$ and a product $\boldsymbol{j}$, there is an edge $\boldsymbol{e}_{\{ij\}}$. Each edge is associated with a set of words (namely, a review) $\boldsymbol{s_{ij}} = \{\boldsymbol{w_{i,j,1}}, \dots, \boldsymbol{w_{i,j,S}}\}$ and a rating $\boldsymbol{r_{ij}}$. The size of the vocabulary, for each product category, is $\boldsymbol{V}$**.** The number of reviews is $\boldsymbol{R}$. We will now present the architecture of our approach.

2.2 Details on the architecture for textual explanation generation

Technically, we embed our model in a neural generative modeling framework, which integrates good properties of probabilistic generative models and neural networks. In particular, we sample the edges and the associated texts as follows:

- For each user i, there is an embedding vector $\boldsymbol{x}_i \in \mathbb{R}^D$ associated, which is sampled from a multivariate Gaussian with zero mean and a diagonal covariance matrix $\mathbf{I}$:
$$\boldsymbol{x}_i \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I})$$
- For all users, we introduce K clusters, and each user cluster k is associated with an embedding vector $\boldsymbol{c}_k \in \mathbb{R}^D$, which is again drawn from a Gaussian:
$$\boldsymbol{c}_k \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I})$$
Here we assume all embedding vectors have the same dimension to avoid complicated notation.
- The user cluster weights θ_i are computed based on the embedding vectors:
$$\theta_{i,k} = \text{sparsegen} - \text{lin}(f(\boldsymbol{x}_i, \boldsymbol{c}_k; \phi): \lambda_u)$$
which specifies the probability of the user i in the cluster k. The function f quantifies the relevance or similarity between the user $\boldsymbol{x}_i$ and the cluster $\boldsymbol{c}_k$. We define the function using a neural network with parameters ϕ, e.g., a MLP with $\boldsymbol{x}_i$ and $\boldsymbol{c}_k$ as inputs and *sparsegen-lin* as output layer. The hyperparameter λ_u represents a regularization term shared among all users. Sparsegen-lin is a controllable extension of sparsemax [16], defined in [17] as:
$$\text{sparsegen} - \text{lin}(\boldsymbol{z}; \lambda) = \text{sparsemax}\left(\frac{\boldsymbol{z}}{1-\lambda}\right)$$
where the coefficient $\lambda < 1$ controls the regularization strength. In particular for $\lambda \to 1^-$, the probability distribution has the minimum support (i.e. *hardmax*) whereas for $\lambda \to -\infty$, the resulting distribution is non-sparse (i.e. *uniform*).
- For each product j we can proceed in an equivalent manner introducing L clusters and generating $\boldsymbol{x}_j, \boldsymbol{c}_\ell$ and $\theta_{l,\ell}$ employing a different neural network g and using hyperparameters ξ and λ_p accordingly.
- We now sample a text associated with an edge $e_{i,j}$. Draw each word $w_{i,j,v}$ in the text as follows:
$$\boldsymbol{z}_{i,v} \sim \text{Categorical}(\theta_i)$$
$$\boldsymbol{z}_{j,v} \sim \text{Categorical}(\theta_j)$$
$$\boldsymbol{w}_{i,j,v} \sim \text{Categorical}(\beta, \boldsymbol{z}_{i,v}, \boldsymbol{z}_{j,v})$$
Where β is a 3D tensor representing the probabilistic patterns among user clusters, product clusters and words. In particular, $\beta_{k,\ell,:}$ specifies a categorical word distribution conditioned on the user cluster k and the product cluster ℓ. It lies in a $(V-1)$-dimensional simplex Δ^{V-1}, i.e. $\sum_{v=1}^{V} \beta_{k,\ell,v} = 1$ and $\beta_{k,\ell,v} > 0$. V denotes the number of words. The parameter $\beta_{k,\ell,v}$ is computed as:
$$\beta_{k,\ell,v} = \text{sparsemax}(\psi(\boldsymbol{c}_k, \boldsymbol{c}_\ell, \boldsymbol{x}_v; \rho))$$

The function ψ defines a neural network with parameters ρ. $\boldsymbol{x}_v$ denotes the pre-trained word vector from Word2Vec [18]. The word sampling is inspired by the topic models. Here we assume every relation and word follow their distributions with distinct parameters computed with functions of the embedding vectors of the involved nodes. Figure 2 depicts a schematic representation of the model.

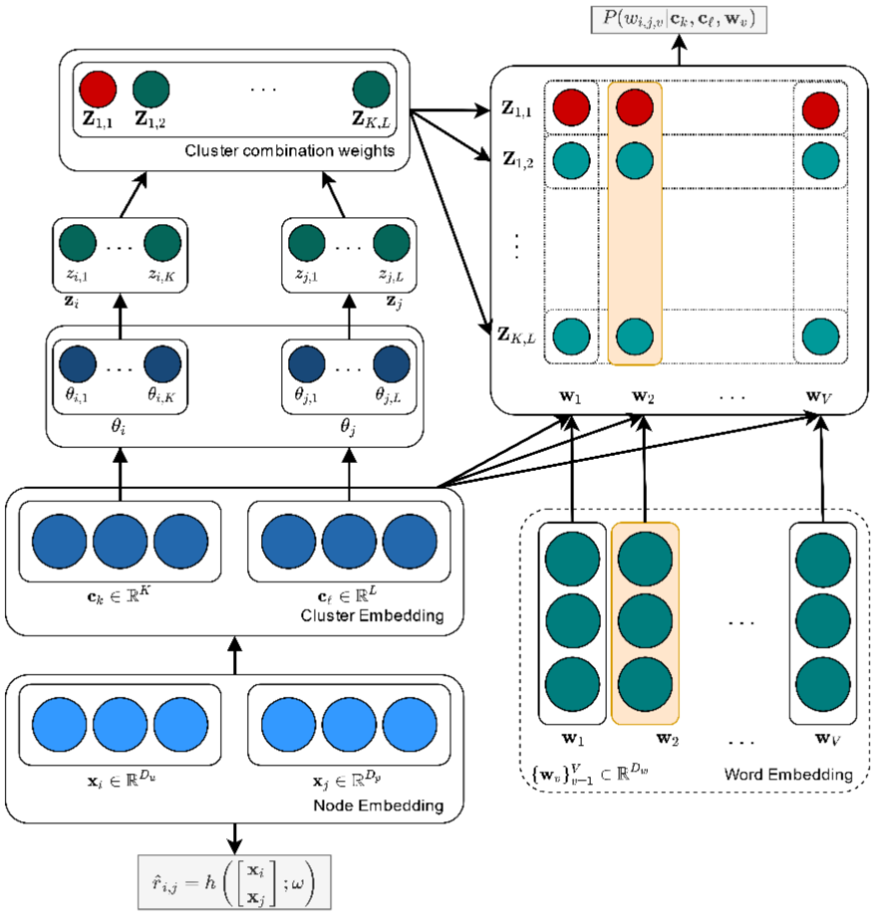

$P(w_{i,j,v}|\mathbf{c}_k,\mathbf{c}_\ell,\mathbf{w}_v)$
$\mathbf{Z}_{1,1}$
$\mathbf{Z}_{1,2}$
$\mathbf{Z}_{K,L}$
Cluster combination weights
$z_{i,1}$
$z_{i,K}$
$\mathbf{z}_i$
$z_{j,1}$
$z_{j,L}$
$\mathbf{z}_j$
$\theta_{i,1}$
$\theta_{i,K}$
θ_i
$\theta_{j,1}$
$\theta_{j,L}$
θ_j
$\mathbf{w}_1$
$\mathbf{w}_2$
$\mathbf{w}_V$
$\mathbf{c}_k \in \mathbb{R}^K$
$\mathbf{c}_\ell \in \mathbb{R}^L$
Cluster Embedding
$\{\mathbf{w}_v\}_{v=1}^V \subset \mathbb{R}^{D_w}$
Word Embedding
$\mathbf{x}_i \in \mathbb{R}^{D_u}$
$\mathbf{x}_j \in \mathbb{R}^{D_p}$
Node Embedding
$\hat{r}_{i,j} = h\left(\begin{bmatrix}\mathbf{x}_i\\ \mathbf{x}_j\end{bmatrix};\omega\right)$

Figure 2 - Schematic view of our model. Dashed boxes represent input (non-trainable) data. The line connections depict the dependencies between the involved variables.

Given the architecture, we can now generate textual explanations of node embeddings. For a node, e.g. a user i, the textual explanation is formulated as a node-specific word distribution $p(\boldsymbol{w}_v|\boldsymbol{x}_i)$ conditioned on its embedding vector $\boldsymbol{x}_i$.
In particular, the probability of a word v to be used to explain the embedding $\boldsymbol{x}_i$ is computed as:

$$p(\boldsymbol{w}_v|\boldsymbol{x}_i) = \frac{1}{L}\sum_{k,\ell} \theta_{i,k}\ \beta_{k,\ell,v}$$

This is a marginal distribution over all possible user and product clusters. Since the target distribution is not related to any specific products, the product clusters are equally distributed, i.e. the term $^1/_L$ in the equation above. Textual explanations for product nodes can be generated in an equivalent manner.

The learning of the node embeddings and the corresponding textual explanations is driven by two learning objectives. The first objective is the log-likelihood of the review corpus. The parameters to be learned include embedding vectors of clusters $\{ \boldsymbol{c}_k\}_{k=1}^{K}$ and $\{ \boldsymbol{c}_\ell\}_{\ell=1}^{L}$, and parameters ϕ, ξ, γ and ρ that define the neural networks f, g and ψ. Thus, the log-likelihood of an edge and the corresponding text is:

$$\begin{aligned}\mathcal{L}_1 = \log p(e_{i,j}\,|\boldsymbol{x}_i, \boldsymbol{x}_j, \gamma) + \\ + \sum_{v=1}^{S} \log \Bigg(\sum_{k=1}^{K} \sum_{\ell=1}^{L} p(\boldsymbol{z}_i = k|\, \boldsymbol{x}_i, \boldsymbol{c}_k, \phi) \\ p\big(\boldsymbol{z}_j = \ell \big|\boldsymbol{x}_j, \boldsymbol{c}_\ell, \xi\big) p\big(\boldsymbol{w}_{i,j,v}\big|\boldsymbol{c}_k, \boldsymbol{c}_\ell, \boldsymbol{x}_v, \rho\big)\Bigg)\end{aligned}$$

where the first term represents the probability that an edge exists between two nodes. This is implicitly included in the computation of the textual information since we suppose that every edge, if exists, has some associated text.

The second objective is the error of the predictions. In our study case, this will be the rating prediction error.

$$\mathcal{L}_2 = \frac{1}{R}\left(\hat{r}_{i,j} - r_{i,j}\right)^2$$

with

$$\hat{r}_{i,j} = h\left(\begin{bmatrix}\boldsymbol{x}_i \\ \boldsymbol{x}_j\end{bmatrix}; \omega\right)$$

where $h(\cdot)$ can be any complex function. In our case, $h(\cdot)$ defines a deep neural network with the concatenation of the node embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as input and ω as hyperparameters. Finally, we can define the complete objective function as:

$$\min_{\Theta} \mathcal{L} = \min_{\Theta} \left(\mathcal{L}_1 + \mu \mathcal{L}_2\right)$$

where Θ represents the parametric space and μ is a hyperparameter to trade-off the importance of the prediction accuracy and the log-likelihood of the corpus.

2.3 Experiment results

We evaluate our method with a popular benchmark dataset: *Amazon Product Data*[1]. The dataset includes millions of product reviews and metadata divided per category collected from May 1999 to July 2014. We focus on the *5-core* version of the data sets, where each user and item has at least 5 associated reviews. The ratings (labels of links between users and items) are integer values between 1 and 5. We use data from 12 categories for our evaluation.

To preprocess the texts associated with the review graphs, we embed the words into a 200-dimensional vector space using Word2Vec [18]. We train the word vectors with the raw reviews to capture the semantic and syntactic structure of the corpus. We normalize the text via the following steps: (a) set the maximum length of a raw review to 300; (b) lower cased letters; (c)

[1] http://jmcauley.ucsd.edu/data/amazon

remove stopwords, numbers and special characters; (d) remove non-existing words using an English vocabulary as filter; (e) lemmatization; (f) remove reviews with just one word.
The textual information shows strong diversity with respect to product category. For example, words frequently occurring in the *Baby* category would not occur in *Office product* reviews. Therefore, the vocabulary selection has been performed independently for each category. Let $\mathcal{C}$ denote the review collection for a given category, we consider each review $\boldsymbol{s}_{ij} \in \mathcal{C}$ as a document. For each word $w \in \boldsymbol{s}_{ij}$, we compute the corresponding *tf-idf* index and extract the top 10% words (at most 10 for longer reviews) with respect to such index. Then, we merge all the top-words extracted from the corpus $\mathcal{C}$ and we sort them with respect to the *tf-idf* score. The first V words in this ranking denote the vocabulary for the given category. Finally, we further filter the reviews to remove the ones without any word belonging to the vocabulary. To tackle the word co-occurrence sparsity problem over short texts, we extract biterms for each document [19]. For a node-related document with v words, the resulting biterm extraction results in $v(v-1)/2$ unordered word-pairs. In this way, we can pattern the textual information by means of biterms co-occurring in the same document. Some statistics of the preprocessed data sets are summarized in Table 1.

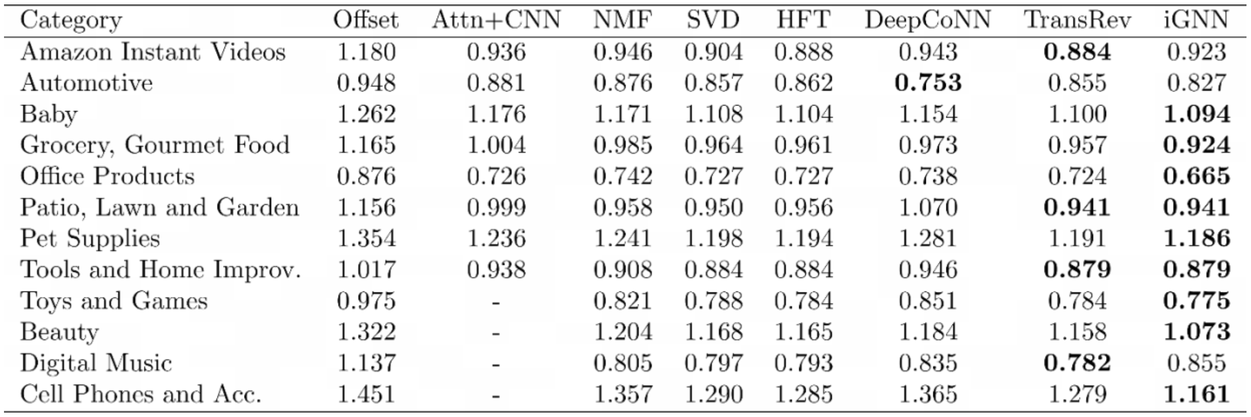
	Reviews	Users	Items
Amazon Instant Videos	36575	4178	1686
Automotive	20303	2277	1835
Baby	157447	17589	7050
Grocery, Gourmet Food	149383	12210	8709
Office Products	53085	4276	2421
Patio, Lawn and Garden	13223	1484	963
Pet Supplies	152705	17079	8510
Tools and Home Improv.	132360	14109	10218
Toys and Games	161775	15541	11919
Beauty	192562	19284	12089
Digital Music	64202	4766	3569
Cell Phones and Acc.	190186	24628	10420

Table 1- Statistics of the preprocessed data sets.

Experimental setting
We set the embedding vector dimensionality $D = 200$ for all users, products and clusters embeddings. We evaluated the robustness of our method to changes in the hyper-parameter D but did not observe any significant performance difference. The number of user clusters $K = 50$ and product clusters $L = 30$; we evaluated the values $[10,20,30,40,50]$ and we observed that, generally, larger values lead to more sparse cluster assignments. We set the coefficients for the *sparsegen-lin* function as $\lambda_u = 0.9$ and $\lambda_p = 0.75$. The function $h(\cdot)$ is a neural network with four hidden fully-connected layers $[128,64,64,32]$. The experiment was run for 200 learning iterations and validated every 2 iterations. A single epoch performs RMSProp with a learning rate set to $2e-6$ and batch size of 256.

Rating prediction results

We compare our method with several baselines considering both factorization-based approaches, like SVD and NMF [20], and review-based approaches (i.e. methods that take advantage of the textual information for improving the rating prediction performance), including HFT [21], DeepCoNN [22], TransRev [23] and Attn+CNN [24]. We also compare our method with a simple baseline that uses the average of the training ratings as prediction. Following previous works, the data is randomly split by reviews into training (80%), validation (10%) and test (10%) sets. We independently repeat each experiment on five different random splits and report the averaged Mean Squared Error (MSE) to quantitatively evaluate the results. Note that we primarily focus on the generation of the textual explanation for node embeddings. This experiment is a direct consequence of how we define the generation of textual explanations and the loss. From (Table 2), we can observe that our approach outperforms other models on the majority of the data sets and gets comparable results on the remaining. This clearly confirms the capacity of our proposed method on learning node embedding that competitively perform on a rating prediction task.



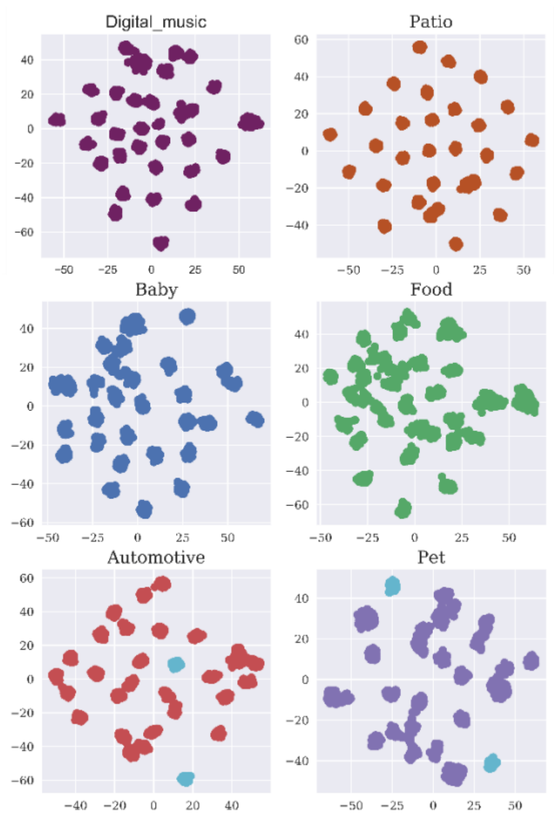
Category	Offset	Attn+CNN	NMF	SVD	HFT	DeepCoNN	TransRev	iGNN
Amazon Instant Videos	1.180	0.936	0.946	0.904	0.888	0.943	**0.884**	0.923
Automotive	0.948	0.881	0.876	0.857	0.862	**0.753**	0.855	0.827
Baby	1.262	1.176	1.171	1.108	1.104	1.154	1.100	**1.094**
Grocery, Gourmet Food	1.165	1.004	0.985	0.964	0.961	0.973	0.957	**0.924**
Office Products	0.876	0.726	0.742	0.727	0.727	0.738	0.724	**0.665**
Patio, Lawn and Garden	1.156	0.999	0.958	0.950	0.956	1.070	**0.941**	**0.941**
Pet Supplies	1.354	1.236	1.241	1.198	1.194	1.281	1.191	**1.186**
Tools and Home Improv.	1.017	0.938	0.908	0.884	0.884	0.946	**0.879**	**0.879**
Toys and Games	0.975	-	0.821	0.788	0.784	0.851	0.784	**0.775**
Beauty	1.322	-	1.204	1.168	1.165	1.184	1.158	**1.073**
Digital Music	1.137	-	0.805	0.797	0.793	0.835	**0.782**	0.855
Cell Phones and Acc.	1.451	-	1.357	1.290	1.285	1.365	1.279	**1.161**

Table 2 Mean Squared Error (MSE) comparison between our method (iGNN) and state-of-the-art approaches.

Analysis of the probabilistic patterns

We now investigate the capacity of our model on generating textual explanations for the learned node vector representations in order to create a human-understandable explanation of them. We first evaluate the probabilistic patterns between user clusters, product clusters and words by analyzing the learned word distributions $\beta_{\cdot,\cdot,v}$. As defined before, β specifies the word distributions for each combination of user and product cluster. For each category, to visualize the learned word-vector distributions, the TSNE method [25] is employed for dimensionality reduction.

Figure 3 illustrates the cluster organization for different categories. One can find that the clusters are well structured and mapped into the 2-dimensional embedding space with different distributions. For further analysis, we select a pair of clusters from two different categories, and we compute the corresponding average word distribution. In theory, one could infer the main *topic* of each cluster looking at the most probable words of the averaged cluster word distribution. (Table 2) reports the most probable words for each of the selected clusters. The results validate our hypothesis since, for each cluster, we can infer the sub-category of interest. For instance, the first cluster in the *Pet Supplies* category refers to the *grooming* sub-category, while the second one focuses on *aquariums*. Thus, the word distributions $\beta_{\cdot,\cdot,v}$ capture the latent structures of the data and help to find the patterns between user and product clusters, enhancing the interpretability of the results. Indeed, knowing the user and product cluster assignments one can find the *sub-categories* highly correlated with the given items.



Automotive		Pet Supplies	
Cluster 1	Cluster 2	Cluster 1	Cluster 2
battery	trailer	work	filter
charge	power	hair	fish
charger	gas	brush	water
power	away	look	gallon
plug	compressor	thing	plant
code	large	fur	heater
cord	pressure	smell	turtle
lead	jeep	quality	flow
transmission	price	wet	clean
phone	guy	stuff	pump
volt	hole	comb	algae
change	weight	shed	shrimp
adapter	fast	groom	work
smell	fuel	flea	gravel
connect	space	shampoo	tube
electronics	*performances*	*grooming*	*aquariums*

Figure 3 – Cluster organization of the word-vector distribution $\beta_{\cdot,\cdot,v}$ for different product categories. The cluster analyzed in the table are highlighted in cyan.

Evaluation of generated textual explanations

To investigate whether our model is able to generate textual explanations as node-specific word distributions, we attempt to provide both quantitative and qualitative evaluation of the node-related explanations. As reported in [26], lacking common metrics for interpretability quality is problematic to the research community to make progress in this area and, in an unsupervised setting, the problem is even more clear. Indeed, it is common for researchers to simply rely on the human visualization of the results, e.g. employing attention mechanisms or heat maps, in order to make the results human-explainable [6]. However, although our case study can be evaluated by just relying on human perception of the highlighted relevant words, we try to introduce some metrics that could further help on assessing the quality of the results. To visualize and evaluate the correlation between the generated word distributions and the node-related words in the data, we proceed as follows. Given a sampled node, we first extract the *top-15* words in the generated word distribution, i.e. the 15 words having the highest probability in the distribution. Second, we extract

the set of words associated with this specific node in the data. Let denote with A and B respectively, these two sets.
The Jaccard similarity is used to measure similarity between two sets of words A and B, which is defined as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

To further capture the semantic similarity of the two sets, we integrate the Word Mover's Distance (WDM), introduced by [27]; this metric takes into account the word similarities in the word embedding space and computes the minimum distance that the words in text A need to *travel* in the semantic space to reach the words in text B. Since Jaccard and WDM have different scales and behaviors, we apply MinMaxScaler to Jaccard, and transform WDM as follows:

$$t - WDM(A,B) \; = \; 1 \; - \; \left(\frac{d_i - \min\,(\boldsymbol{d})}{\max(\boldsymbol{d}) - \min\,(\boldsymbol{d})}\right)$$

where d_i is the distance between the sets A and B for a node i, and $\boldsymbol{d}$ represents all the distances between the two sets for each node in the graph. In this way, the transformed distance score is in the range [0,1] and, opposite to the definition of semantic distance, the higher the value the closer are sets A and B.

We investigate the quality of the generated textual explanations by computing these scores for different data sets. Figure 4 illustrates the distribution of values of these measures across the nodes in the data sets.

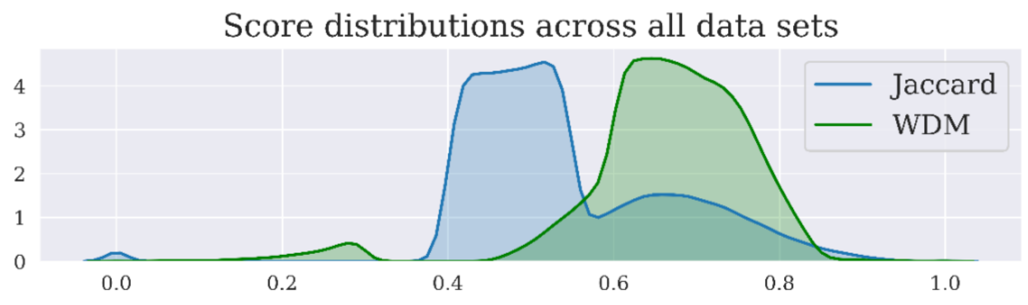
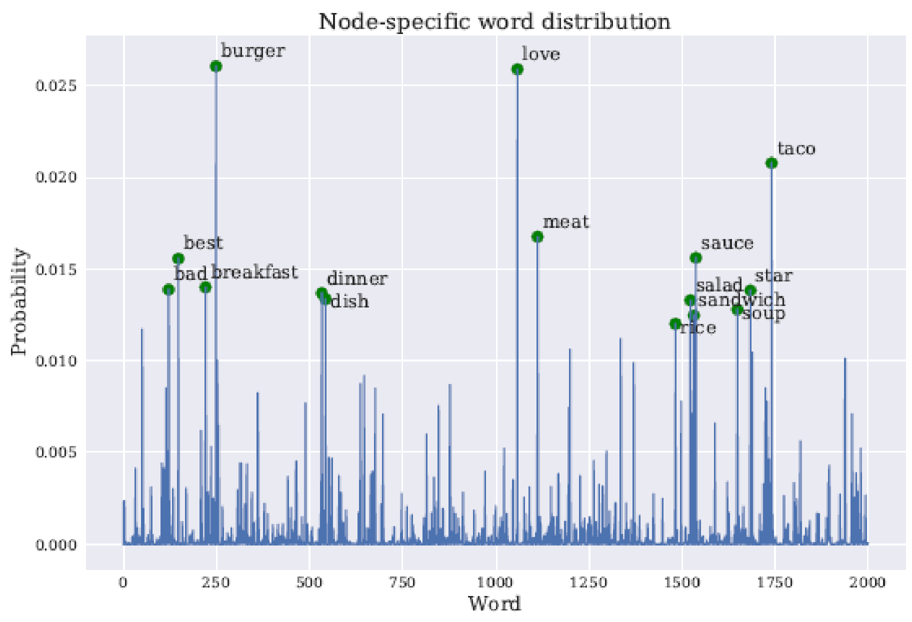


Figure 4 - Evaluation of the metric distributions across all data sets.

The Jaccard values mostly vary in the range $[0.4 - 0.7]$, while the WDM scores are highly concentrated in the range $[0.6 - 0.8]$. The lower Jaccard scores do not affect the performances of the model, instead, confirms that our model generates textual explanations that are not redundant as would have been with too high similarity scores. Indeed, as written in [6], two sets of words can be semantically similar even with low lexical overlapping. Moreover, this proves that our approach is not solely based on lexical similarity, but takes into account the semantic similarity of the words generating textual explanations that are not too similar to the original data.

Figure 5 shows an example of the generated word distribution for a sampled node.



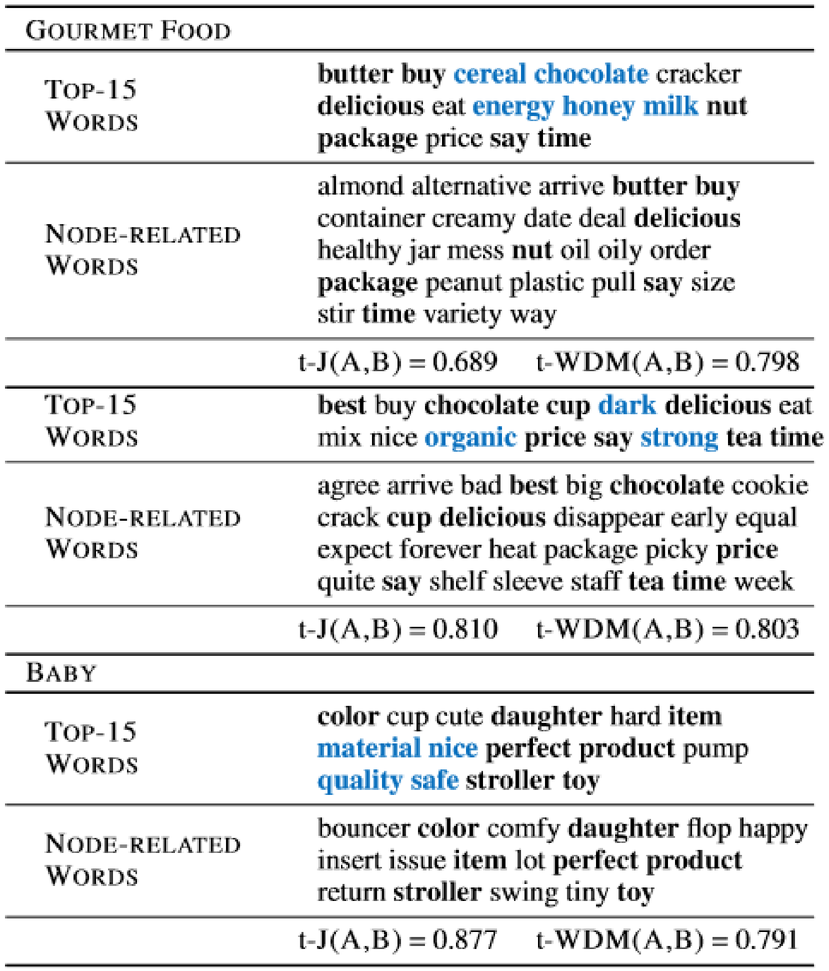
Top-15 Words	bad **best** **breakfast** **burger** dinner dish **love** **meat** **rice** **salad** **sandwich** **sauce** soup star **taco**	
Node-related Words	ambiance authentic **best** **breakfast** choice cilantro decision guacamole **love** **meat** pas plain satisfied **sauce** spacious staple **taco**	
	t-J(A,B) = 0.796	t-WDM(A,B) = 0.748

Figure 5 – Interpretability case study on a random node. The figure depicts the node-specific word distribution; the 15 highest probabilities are highlighted by green points. TOP-15 WORDS and NODE-RELATED WORDS refer to the sets A and B defined above. Black bold represents the overlapping words; blue bold highlights words that may explain further characteristics of the analyzed node.

We further evaluate the results on the *Baby*, *Gourmet Food*, and *Pet Supplies* categories. The example nodes in Table 3 demonstrate the effectiveness of our model to capture the most probable words associated with a given node. In particular, not only direct correspondences with the words in the data but also highly related words.

For example, looking at the sampled node from the *Baby* category, we can observe that the generated explanation might be used to better capture further characteristics of the given node. Specifically, it looks like the node is related to toys and strollers and, analyzing the blue bold highlighted words, we can infer that the items should also be safe and made with good quality materials. Additionally, we can notice the impact of the scores on the quality of the generated word distributions. The first two examples, taken from the *Gourmet Food* data set, have different Jaccard scores while similar distances. Nevertheless, the first example looks semantically good. This shows the importance of considering the word vector representations during the training phase since they can *push* the probabilities to words semantically closer.

We can also observe that: (a) the model does not take into account the polarity of the words. An extension could be achieved by exploiting the sentiment (or rating) information to get *positive*/*negative* word distributions; (b) the quality of vocabulary could be improved by using more sophisticated techniques that might potentially lead to better performances.



GOURMET FOOD	
TOP-15 WORDS	**butter buy** **cereal chocolate** cracker **delicious** eat **energy honey milk** **nut** **package** price **say time**
NODE-RELATED WORDS	almond alternative arrive **butter buy** container creamy date deal **delicious** healthy jar mess **nut** oil oily order **package** peanut plastic pull **say** size stir **time** variety way
	t-J(A,B) = 0.689 t-WDM(A,B) = 0.798
TOP-15 WORDS	**best** buy **chocolate cup** **dark** **delicious** eat mix nice **organic** **price say** **strong** **tea time**
NODE-RELATED WORDS	agree arrive bad **best** big **chocolate** cookie crack **cup delicious** disappear early equal expect forever heat package picky **price** quite **say** shelf sleeve staff **tea time** week
	t-J(A,B) = 0.810 t-WDM(A,B) = 0.803
BABY	
TOP-15 WORDS	**color** cup cute **daughter** hard **item** **material nice** **perfect product** pump **quality safe** **stroller toy**
NODE-RELATED WORDS	bouncer **color** comfy **daughter** flop happy insert issue **item** lot **perfect product** return **stroller** swing tiny **toy**
	t-J(A,B) = 0.877 t-WDM(A,B) = 0.791

Table 3 – Evaluation of retrieved nodes from different data sets. Black bolds represents the matching words; blue bold highlights words that may explain further characteristics of the node.

3 Probabilistic framework for topographic organization of user and product latent classes based on product reviews

The availability of large collections of textual data (e.g. product reviews, social media posts) has driven researchers to focus on developing text-based methods for recommendation. Indeed, corpora of textual documents contain a wealth of information. On the one hand, they may help users to make decisions that are more aware. On the other hand, they may be used to improve the predictive performance of recommendation systems. In recent years, many approaches have tried to extend and improve recommendations models by leveraging the textual information. However, most of the proposed models mainly focus on predictive performances, while less attention has been paid on understanding the nature of the reviews and on developing visualization components to enhance the interpretability capacity of the models. When available, the visualization component is not model-based. In many cases, approaches like t-SNE [25] are employed for visualizing high-dimensional representations into a 2-dimensional space. However, the resulting plot is just a projection and does not reflect any consistent visualization-driven model formulation of the underlying patterns. Thus, the output could be misleading and erroneously evaluated. In this work, we present a probabilistic framework for the topographic organization of review data. Differently from previous works mainly focused on the rating patterns [28], we impose a double two-dimensional topological organization based on the textual information. As a result, latent classes of users and products are organized on two different square grids. The visualization in a two-dimensional grid of word patterns provide an accessible way for exploring latent features of users and products using complicated and large amounts of textual data. Moreover, the probabilistic assumptions of our system enable us to analyze the extracted information in a statistical way.

3.1 Details on the proposed probabilistic model

Assume a collection of N users $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$, M products $\mathcal{P} = \{p_1, p_2, \dots, p_M\}$ and V words $\mathcal{V} = \{w_1, w_2, \dots, w_V\}$. The data $\mathcal{D}$ is a collection of R triples $\mathcal{D} = \{(u^i, p^i, r^i)\}_{i=1}^{R}$, each triple identifying the user $u^i \in \mathcal{U}$ writing a review r^i on product $p^i \in \mathcal{P}$. The review r^i is a multi-set of words from $\mathcal{V}$, i.e.

$$r^i = \left(w_1^i, w_2^i, \dots, w_{S_i}^i\right), \qquad w_i^j \in \mathcal{V}$$

The latent variables $\boldsymbol{z}_u \in \{1, \dots, K\}$ and $\boldsymbol{z}_p \in \{1, \dots, L\}$ represent *abstract classes* of users and products. Given a review i, the probability of a word $w_j \in r^i$ is modeled as:

$$P\left(w_j^i \middle| u^i, p^i\right) = \sum_{k}^{K} \sum_{\ell}^{L} P\left(w_j^i \middle| \boldsymbol{z}_u = k, \boldsymbol{z}_p = \ell\right) P(\boldsymbol{z}_u = k | u^i) P(\boldsymbol{z}_p = \ell | p^i)$$

Introducing the topological organization

A grid topology is introduced into the latent space via the channel noise methodology. The channel noise is expressed through the *neighborhood function*:

$$P(\boldsymbol{y}|\boldsymbol{z}) = \frac{\exp\left(\frac{-|\boldsymbol{z}-\boldsymbol{y}|^2}{2\sigma^2}\right)}{\sum_{\boldsymbol{y}'}\exp\left(\frac{-|\boldsymbol{z}-\boldsymbol{y}'|^2}{2\sigma^2}\right)}$$

For the latent classes $\boldsymbol{y}$ and $\boldsymbol{z}$ lying close to each other on the grid, the probability of corrupting one into the other is high. The parameter $\sigma > 0$ represents the *specificity* of the topological neighborhood for class $\boldsymbol{z}$; low values of σ correspond to sharply peaked localized transition probabilities, while larger values of σ induce general broad neighborhoods spanning large areas of the latent grid.

For each user $u \in \mathcal{U}$, we proceed as follows:

1. Randomly generate a latent class index $\boldsymbol{z}_u = k$ by sampling the user-conditional probability distribution $P(\cdot\,|u)$ on $\boldsymbol{z}_u$.
2. Transmit the class identification $\boldsymbol{z}_u$ through a noisy communication channel, and receive (a possibly different) class index $\boldsymbol{y}_u = k'$.

Where the class index represents a particular point on the grid.

For each product $p \in \mathcal{P}$, we can proceed in an equivalent manner.

The model has now the following form:

$$P\left(\boldsymbol{y}_u = k'\middle|u^i\right) = \sum_{k=1}^{K} P(\boldsymbol{y}_u = k'|\boldsymbol{z}_u = k)P\left(\boldsymbol{z}_u = k\middle|u^i\right)$$

$$P\left(\boldsymbol{y}_p = \ell'\middle|p^i\right) = \sum_{\ell=1}^{L} P\left(\boldsymbol{y}_p = \ell'\middle|\boldsymbol{z}_p = \ell\right)P\left(\boldsymbol{z}_p = \ell\middle|p^i\right)$$

$$P\left(w_j^i\middle|u^i, p^i\right) = \sum_{k}^{K}\sum_{\ell}^{L} P\left(w_j^i\middle|\boldsymbol{y}_u = k', \boldsymbol{y}_p = \ell'\right)P\left(\boldsymbol{y}_u = k'\middle|u^i\right)P\left(\boldsymbol{y}_p = \ell'\middle|p^i\right)$$

Inference and learning

Assuming independent data items in $\mathcal{D}$, the log-likelihood of the model is:

$$\mathcal{L} = \sum_{i=1}^{R} \log P(r^i|u^i, p^i)$$

We will consider a simple model assuming independence of the words appearing in the review:

$$P\left(r^i\middle|u^i, p^i\right) = \prod_{j=1}^{S_i} P\left(w_j^i\middle|u^i, p^i\right)$$

Hence, the log-likelihood reads:

$$\mathcal{L} = \sum_{i=1}^{R}\sum_{j=1}^{S_i} \log P\left(w_j^i\middle|u^i, p^i\right)$$

Finally, we obtain:

$$\mathcal{L} = \sum_{i=1}^{R}\sum_{j=1}^{S_i} \log\left[\sum_{k'=1}^{K}\sum_{\ell'=1}^{L} P\left(w_j^i \middle| \boldsymbol{y}_u = k', \boldsymbol{y}_p = \ell'\right) \right.$$
$$\left. \sum_{k=1}^{K} P(\boldsymbol{y}_u = k' | \boldsymbol{z}_u = k) P\left(\boldsymbol{z}_u = k \middle| u^i\right) \sum_{\ell=1}^{L} P\left(\boldsymbol{y}_p = \ell' \middle| \boldsymbol{z}_p = \ell\right) P\left(\boldsymbol{z}_p = \ell \middle| p^i\right)\right]$$

For training, we use Expectation-Maximization (EM) algorithm. The EM algorithm is a standard iterative approach for soft assignment of points to latent classes. In this way, through maximum likelihood estimation (MLE), each item has a probability of belonging to a specific latent class. It runs iteratively two steps, Expectation (E) and Maximization M), until convergence.

3.1.1 E-step

In the E-step, the algorithm evaluates the current estimates of the model parameters by computing the expected values of the latent variables.
Note that we have two levels of hidden variables. First, given the user and the product they reviewed, we do not know which latent classes $\boldsymbol{z}_u$ and $\boldsymbol{z}_p$ represent the (*user*, *product*) couple when writing the review. Second, we know that the underlying latent classes $\boldsymbol{z}_u$ and $\boldsymbol{z}_p$ may have been disrupted to latent classes $\boldsymbol{y}_u$ and $\boldsymbol{y}_p$ before producing the review, but we do not know their identity. To simplify mathematical notation, we will denote $\boldsymbol{z}_u = k$, $\boldsymbol{z}_p = \ell$, $\boldsymbol{y}_u = k'$ and $\boldsymbol{y}_p = \ell'$ as $\boldsymbol{z}_u^k$, $\boldsymbol{z}_p^\ell$, $\boldsymbol{y}_u^{k'}$ and $\boldsymbol{y}_p^{\ell'}$. In the E-step, the algorithm computes the expected values of latent variables using the current values of model parameters.

$$\hat{P}\left(\boldsymbol{z}_u^k, \boldsymbol{z}_p^\ell \middle| w, u, p\right) =$$
$$= \frac{P(\boldsymbol{z}_u^k|u) P\left(\boldsymbol{z}_p^\ell \middle| p\right) \sum_{k'}\sum_{\ell'} P(w|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}) P(\boldsymbol{y}_u^{k'}|\boldsymbol{z}_u^k) P(\boldsymbol{y}_p^{\ell'}|\boldsymbol{z}_p^\ell)}{\sum_{k''}\sum_{\ell''} P(\boldsymbol{z}_u^{k''}|u) P\left(\boldsymbol{z}_p^{\ell''} \middle| p\right) \sum_{k'}\sum_{\ell'} P(w|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}) P(\boldsymbol{y}_u^{k'}|\boldsymbol{z}_u^{k''}) P(\boldsymbol{y}_p^{\ell'}|\boldsymbol{z}_p^{\ell''})}$$

$$\hat{P}\left(\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'} \middle| w, u, p\right) =$$
$$= \frac{P(w|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}) \sum_k P(\boldsymbol{y}_u^{k'}|\boldsymbol{z}_u^k) P(\boldsymbol{z}_u^k|u) \sum_\ell P(\boldsymbol{y}_p^{\ell'}|\boldsymbol{z}_p^\ell) P\left(\boldsymbol{z}_p^\ell \middle| p\right)}{\sum_{k''}\sum_{\ell''} P(w|\boldsymbol{y}_u^{k''}, \boldsymbol{y}_p^{\ell''}) \sum_k P(\boldsymbol{y}_u^{k''}|\boldsymbol{z}_u^k) P(\boldsymbol{z}_u^k|u) \sum_\ell P(\boldsymbol{y}_p^{\ell''}|\boldsymbol{z}_p^\ell) P\left(\boldsymbol{z}_p^\ell \middle| p\right)}$$

3.1.2 M-step

In the M-step, the algorithm maximizes the expectation computed in the E-step by re-estimating the model parameters. To do so, we need to specify the distributions for $P(\boldsymbol{z}_u^k|u)$, $P\left(\boldsymbol{z}_p^\ell \middle| p\right)$ and $P\left(w \middle| \boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right)$. It looks natural to model these parameters as multinomial distributions.

Thus, we assume:

$$P(\boldsymbol{z}_u|u) \sim \text{Multinomial}$$
$$P(\boldsymbol{z}_p|p) \sim \text{Multinomial}$$
$$P(w|u,p) \sim \text{Multinomial}$$

The updated equation for $P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right)$ is:

$$P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right) = \frac{\sum_{(u,p)\in\boldsymbol{\mathcal{B}}(w)} \hat{P}\left(\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\middle|w,u,p\right)}{\sum_{w'}\sum_{(u,p)\in\boldsymbol{\mathcal{B}}(w')} \hat{P}\left(\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\middle|w',u,p\right)}$$

where $\boldsymbol{\mathcal{B}}(w)$ is the set of (*user*, *product*) couples associated with the word w.

Denoting the set of words used by user u to review product p by $\boldsymbol{\mathcal{W}}(u,p)$ we obtain the updated equations for $P(\boldsymbol{z}_u^k|u)$and $P\left(\boldsymbol{z}_p^\ell\middle|p\right)$ as:

$$P(\boldsymbol{z}_u^k|u) = \frac{\sum_p \sum_{w\in\boldsymbol{\mathcal{W}}(u,p)} \sum_\ell \hat{P}\left(\boldsymbol{z}_u^k, \boldsymbol{z}_p^\ell\middle|w,u,p\right)}{\sum_p |\boldsymbol{\mathcal{W}}(u,p)|}$$

$$P\left(\boldsymbol{z}_p^\ell\middle|p\right) = \frac{\sum_u \sum_{w\in\boldsymbol{\mathcal{W}}(u,p)} \sum_k \hat{P}\left(\boldsymbol{z}_u^k, \boldsymbol{z}_p^\ell\middle|w,u,p\right)}{\sum_u |\boldsymbol{\mathcal{W}}(u,p)|}$$

3.1.3 Topographic initialization with SOM

The successful application of the EM algorithm depends on the initial position in the parameter space, since the algorithm is strongly sensitive to parameter initialization. We follow an approach similar to the one presented in [28]. Differently from their work, where the SOM was run on a dataset of user ratings, we run two different instances of SOM (one for users and one for products) using the review information. The number of nodes in SOM is equal to the number of latent classes, i.e. K for users and L for products. We denote by $\mathcal{V}$ the vocabulary set. For each user u, there is a $|\mathcal{V}|$-dimensional vector $\boldsymbol{v}$ associated. Each vector dimension represents a word w in the vocabulary and the corresponding value is the term frequency, i.e. how many times the user has written a review using that word. The analogous reasoning is valid for products. In the latter case, each vector dimension represents how many times a word w was used for reviewing the product p.
After training the two instances of SOM, the user and product conditional latent priors, i.e. $P(\boldsymbol{z}_u^k|u)$ and $P\left(\boldsymbol{z}_p^\ell\middle|p\right)$ respectively, are computed as follows. If the user u belongs to the cluster defined by the node $k \in \mathcal{Z}_u = \{1, \dots, K\}$ of the SOM, then we *softened* the binary membership using the following transformation:

$$P(\boldsymbol{z}_u^k|u) = \begin{cases} A & if\ u \in k \\ (1-A)/((K-1) & if\ u \notin k \end{cases}$$

In this way, if the user u belongs to the cluster k, $P(\boldsymbol{z}_u^k|u)$ is $B > 1$ times higher than $P(\boldsymbol{z}_u^{k'}|u)$ for all the other $k' \in \mathcal{Z}_u$. Thus, $A = B/(K-1+B)$. The product conditional latent prior can be generated in an equivalent manner by changing the parameters accordingly.
The empirical distribution for word patterns $P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right)$ is then computed as: (a) to introduce the topology, we follow the steps described in the corresponding subsection to get (possibly corrupted) class indices $\boldsymbol{y}_u^{k'}$ and $\boldsymbol{y}_p^{\ell'}$ for all users u and products p; (b) we denote with $N(w, k', l')$ the number of times the word w has been used by users belonging to the latent class k' to review products that belong to the latent class l'. The empirical distribution is estimated as:

$$P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right) = \frac{N(w, k', l') + \mathrm{m}}{mV + \sum_{w' \in \mathcal{V}} N(w', k', l')}$$

Due to sparseness, we apply a Laplace correction for smoothing the empirical estimates. The parameter m is a positive number and, in this case, $m=1$. For further details, we refer the reader to \cite{Polcicova2004}.

3.2 Experimental analysis

After running the EM algorithm, we have the estimates of the conditional distributions $P(\boldsymbol{z}_u^k|u)$ and \$ $P\left(\boldsymbol{z}_p^{\ell}\middle|p\right)$)\$ for all users and products in our dataset. These quantities represent the probability assignments of the items to their respective latent classes.
As in 2.3, we evaluate our method with the popular *Amazon Product Data* dataset. Again, we focus on the *5-core* version of the data sets and we follow a similar preprocessing procedure. Given the model parameters estimated in Section 3.1, we can analyze user and product latent class word distributions. For each latent class, we can compute the corresponding word distribution and get an insight of the user preferences or product characteristics. The word distribution for each user latent class is:

$$P\left(w, \boldsymbol{y}_p^{\ell'} | \boldsymbol{y}_u^{k'}\right) = P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right) P\left(\boldsymbol{y}_p^{\ell'}\right) \Rightarrow \frac{1}{L} \sum_{\ell'=1}^{L} P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right)$$

since the target distribution is not related to any product, this is a marginal distribution over all the product latent classes (i.e. the term $1/L$ =in the equation).
Equivalently, the word-distribution for each product latent class can be derived as follows:

$$P\left(w, \boldsymbol{y}_u^{k'} | \boldsymbol{y}_p^{\ell'}\right) = P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right) P\left(\boldsymbol{y}_u^{k'}\right) \Rightarrow \frac{1}{K} \sum_{k'=1}^{L} P\left(w\middle|\boldsymbol{y}_u^{k'}, \boldsymbol{y}_p^{\ell'}\right)$$

The visualization of the most probable words associated with each latent class help us to evaluate the results. For visualization purposes, for each latent class word distribution, we extract the 10 most probable words. In theory, from this word list, one should be able to infer the main *topic* of the associated latent class. Additionally, given the assumptions of our model, we should be able to recognize an organization of the latent classes induced by the grid topology. Classes with similar topics should lie close to each other in the grid.

In Table 4, we can see an example of the latent organization of product classes for the *Automotive* category.

light	light	blade	wax
bulb	fit	wiper	kit
bright	lead	great	product
stock	great	fit	great
white	good	good	wiper
brighter	bright	product	water
fit	price	well	snow
price	well	rain	blade
lead	order	brand	wipe
replacement	horn	jeep	rain
filter	great	product	product
oil	good	hose	paint
change	fit	great	great
fit	tool	good	pad
price	price	tank	clay
fuel	product	fit	good
socket	code	well	wax
mile	well	try	polish
wrench	purchase	say	try
good	say	quality	water
oil	good	product	product
change	great	great	jack
fluid	product	good	great
pump	mount	cover	wash
drain	item	strap	shine
price	quality	lock	spray
leak	fit	trailer	trailer
transmission	price	hitch	good
good	door	door	lift
great	hitch	step	wax
battery	plug	wheel	towel
charge	tender	leather	wash
power	battery	brush	water
charger	great	cleaner	gauge
plug	product	seat	pressure
device	cable	great	cloth
motorcycle	motorcycle	level	cleaning
phone	good	product	accurate
compressor	spark	trailer	valve
cord	winter	good	great

Table 4 - Product latent class word-distributions - Automotive category.

There are clear patterns of the most probable words associated with latent classes. Note the topological organization of the latent classes: words patterns are similar for classes that are close to each other. For example, adjacent classes at the top left of the grid refer to *lighting accessories*, while going down to the bottom left we can induce that the latent classes refer to *electrical* and *oil system tools*. On the right part of the grid, instead, we have latent classes referring to *cleaning accessories*. Knowing the assignment probabilities of a sampled item, we can analyze the most probable latent classes and evaluate the corresponding word distributions to get an insight about the item characteristics, making the results more interpretable.

4 Discussion and future works

As anticipated, user-generated data include different types than just the textual one. For this reason, to comprehensively investigate real-world scenarios, we want to employ other types of data that can model different real-world situations. In particular, we will focus our attention on time series and, more specifically, on event point processes since they can naturally model many real-world dynamics. In the next subsections, we will review the current state-of-the-art in this area and we will investigate potentially new ways for improving event sequence modeling.

4.1 State of the art and problem statement

Sequences of discrete events in continuous time can naturally model many real-world scenarios (e.g. medical events, customer behaviors, social media actions). In this scenario, the events are also usually correlated. A single event, or a pattern of events, may help to cause (*excitation*) or prevent (*inhibition*) future events. Discover such patterns can help us describe complicated dynamics in real-world applications and to predict which type of event will happen next at what time. To this end, the intensity of the different types k of event at the time t is modeled using the so-called *intensity function* $\lambda_k(t)$. From the basic model for event streams, the Poisson process (which assumes that events occur independently of one another), many steps have been made to model the complex real-world patterns using $\lambda_k(t)$. In [29], the authors propose a neural self-modulating multivariate process where the intensities of multiple event types evolve according to a new continuous-time LSTM. In this way, they overcome the unrealistic assumptions of the Hawkes processes (where the probability of future events can be raised by the past events only in a positive, additive way) by capturing effects that can be even subtractive.

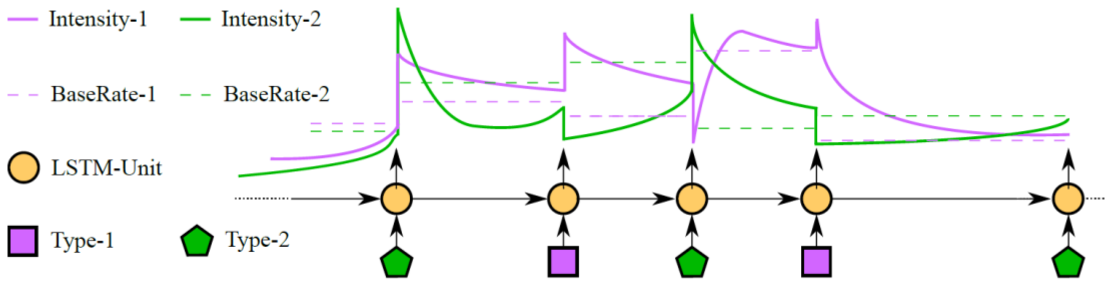


Figure 6 – Example of event time series (two types of event) with inhibition/exhibition effects taken from [29].

Following similar directions, other works (e.g. [30] [31] [32]) attempted to model this type of event streams. In [30], similarly to [29], the intensity function is defined as a nonlinear function of the history, and use a recurrent neural network to automatically learn a representation of influences from the event history. In [32], the authors propose a nonparametric model based on Gaussian processes. However, none of them attempt to generate interpretable models/predictions. Inspired by this limitation and by the works on interpretable time series forecasting [33] [34] [35] [36] we will attempt to develop an interpretable model for multivariate temporal time series. Modeling this type of event streams seems natural and useful in many applied domains, and we believe is important to work in this direction to develop interpretable frameworks in this scenario.

With the advent of deep learning, as written in [34], CNNs have been considered state-of-the-art for multivariate time series classification due to their ability to learn meaningful representations from sequential data without the need for manual feature engineering (e.g. [29] [30]). However, as many machine learning methods, these networks are considered as black box models. The authors propose a method for achieving explainable deep neural network predictions by using a gradient-based approach for generating *saliency maps*. However, these methods are mostly used in vision and language tasks, and their applications to time series data is relatively unexplored [36]. Based on the extensive experiments of the authors, [36] reports that saliency methods tend to fail providing high-quality interpretation in time series data, and proposes a new approach to improve the quality of the saliency methods. In detail, they propose a *two-step temporal saliency rescaling (TSR)* that works as follows: (i) they first calculate the *time-relevance score* for each time step; (ii) they calculate a *feature-relevance score* for each feature in each time step having a time-relevance score higher than a certain threshold. Figure 7 shows the generated saliency maps when applying the two-step TSR approach.

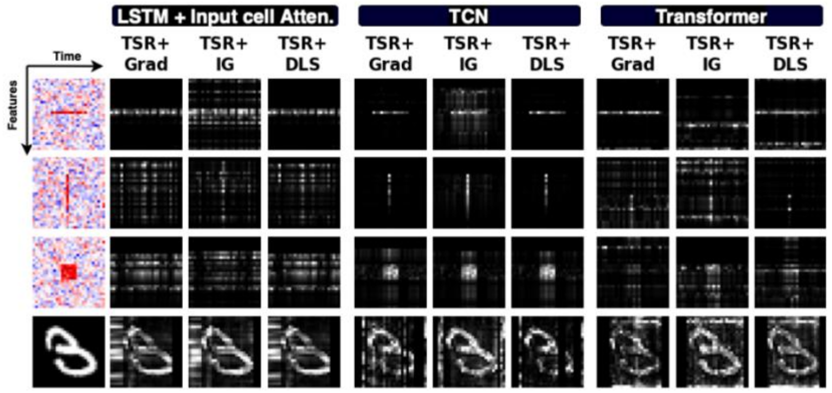


Figure 7 - Example of saliency maps on synthetic data and time series MNIST produced by the approach proposed in [36].

4.2 Future directions

Differently from the state-of-the-art approaches mentioned above, in [37] the authors propose a framework for modeling the intensity function of point processes involving also dense numerical time series via RNNs. Indeed, in real-world applications, the event time series are usually associated with observable numerical time series that often reveal the background status of the system. For example, in medical applications, we could have patients' monitored measurements (e.g. heart rate, blood pressure, glycaemia) that could help us predicting the next event (e.g. a complication). In online retailers and social networks, we could have additional information about users and items that could help unveil the next possible action, e.g. an online purchase. Following a similar approach, we also aim to integrate the available additional information (in the form of numerical time series) during the learning phase for improving the predictive performances of our model. Additionally, inspired by the works on interpretable time series forecasting, we aim at developing an interpretable model for event point processes. In this way, we will be able to highlight the importance of the input features in model predictions for each event and we would be able to understand and explain the dynamics of the involved system. As direct consequence, the better understanding of the underlying system should also improve the prediction task, creating an understandable learning system.

5 Conclusions

This report introduces the essential components for interpretable modeling of user-generated data. Given the nature of user-generated data, we can categorize our contribution into two main branches: the first one where we exploit the user-generated textual information (e.g. reviews), and the second one where we focus our attention on user-associated time series. We provided details on our developed approaches, ongoing works, and future ideas. We reported the results and introduced potential applications in real-world applications. The results demonstrate the ability of our models of exploiting the available user-generated data in favor of interpretability while still performing well on prediction tasks.

In section 2, we present an approach that uses the extra textual information associated with the nodes to learn human-understandable explanations for the corresponding latent representations. To strengthen the interpretability of the results, the model learns simultaneously node embeddings and textual explanations during the training phase. Additionally, the introduction of node cluster embedding enables the model to learn the patterns among nodes, ratings and textual information. In this way, we learn the discrete structures of the graph data and the text-based explanations in an elegant way. Finally, our model is flexible since it can be included with different learning tasks. In our case, since we have the rating information, we perform a rating prediction task, but it can be different. The promising results in the qualitative analysis shows the capacity of the model on generating human-understandable explanation while competitively performing well on a prediction task. For future work, it would be interesting to extend the work by integrating the polarity of the words and employing the interpretation for downstream tasks, such as human-understandable recommendation.

In section 3, we present a probabilistic framework for the organization of user and latent classes based on the review information. The results clearly demonstrate the ability of our proposed model to interpret the latent classes of users and products and to organize them in a structured way. Potentially, to further investigate the results, one could select a user latent class, and retrieve the corresponding product class word distributions to see how the word distributions change when fixing a specific user latent class (and vice versa, starting from a product latent class). We are currently working on an extension of the work, in order to make our probabilistic model generative. In this way, given an unknown user that reviewed a product in our dataset, we would be able to learn the user latent class assignments given the review text, using the posterior distribution.
Most of the works in recommendation focuses on modeling the user preferences by means of the rating information. However, the most nuanced and sophisticated medium to express our feelings is our language [38]. For this reason, we believe is important to organize the review information by means of two-dimensional grids for user and product latent classes. There are several motivations for this: first, there is a strong correlation between reviews and ratings; the review text is indeed the explanation to justify a given rating. Second, the visualization in a two-dimensional grid of word patterns provide an accessible way for exploring latent features of users and products using complicated and large amounts of textual data. Last, since the lower-dimensional latent representations are a good approximation of the textual input space and, in turn, this is strictly related to ratings, we believe that we could achieve good results in the rating prediction using the

resulting latent features as input. Further experiments will follow in this direction. Finally, to facilitate the investigation of the results, we aim at creating an interactive visualization tool that could help the experts to retrieve interesting patterns of the items in order to analyze the results autonomously.

In both cases, as said at the beginning, the experimental setting is an e-commerce system, but the model could be integrated with other real-world scenarios, as long as some textual information is available. For future work, in line with the objectives of the ECOLE project, we aim at integrating our findings into a synergic multi-domain optimization framework.

In conclusion, the challenge presented by the interpretability of machine learning methods (in particular, deep learning ones) is a barrier preventing their serious adoption in real-world applications [36]. Many critical applications involve textual data and time series, but using just accurate and fast algorithms is not sufficient in these cases. To support humans in a cooperative process with AI, we need to develop models and techniques that are human-understandable. Our work aims to contribute on solving this problem, and the results of our findings will help to increase human trustiness and to ease the application of the proposed models where possible. In this way, filling the gap between humans and AI, the evaluation process of AI-based applications will be faster, easier and more reliable.

6 Bibliography

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A survey of methods for explaining black box models, ACM computing surveys (CSUR), 51(5), 1-42, 2018.

[2] P. Koh and P. Liang, Understanding black-box predictions via influence functions, arXiv preprint arXiv:1703.04730, 2017.

[3] B. Liang, H. Li, M. Su, P. Bian, X. Li and W. and Shi, Deep text classification can be fooled, arXiv preprint arXiv:1704.08006, 2017.

[4] A. Nguyen, J. Yosinski and J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 427-436), 2015.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199,* 2013.

[6] I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. Gershman and F. Doshi-Velez, "An evaluation of the human-interpretability of explanation," *arXiv preprint arXiv:1902.00006,* 2019.

[7] M. Ribeiro, S. Singh and C. Guestrin, "Why Should I Trust You: Explaining the Predictions of Any Classifier," in *Proceedings of KDD*, 2016.

[8] S. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions," in *NIPS*, 2017.

[9] F. Wang and C. Rudin, "Falling rule lists," in *Artificial Intelligence and Statistics*, 2015.

[10] H. Lakkaraju, S. H. Bach and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[12] M. Sundararajan, A. Taly and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, 2017.

[13] C. Lawrence, T. Sztyler and M. Niepert, "Explaining Neural Matrix Factorization with Gradient Rollback," *ArXiv,* vol. abs/2010.05516, 2020.

[14] S. Al-Sayouri, E. Gujral, D. Koutra, E. E. Papalexakis and S. S. Lam, "t-pine: Tensor-based predictable and interpretable node embeddings," *Social Network Analysis and Mining,* vol. 10, p. 1–11, 2020.

[15] C. T. Duong, Q. V. H. Nguyen and K. Aberer, "Interpretable node embeddings with mincut loss," 2019.

[16] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *International Conference on Machine Learning*, 2016.

[17] A. Laha, S. A. Chemmengath, P. Agrawal, M. Khapra, K. Sankaranarayanan and H. G. Ramaswamy, "On controllable sparse alternatives to softmax," in *Advances in Neural Information Processing Systems*, 2018.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013.

[19] X. Yan, J. Guo, Y. Lan and X. Cheng, "A biterm topic model for short texts," in *Proceedings of the 22nd international conference on World Wide Web*, 2013.

[20] Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer,* vol. 42, p. 30–37, 2009.

[21] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013.

[22] L. Zheng, V. Noroozi and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017.

[23] A. Garcia-Duran, R. Gonzalez, D. Onoro-Rubio, M. Niepert and H. Li, "Transrev: Modeling reviews as translations from users to items," *arXiv preprint arXiv:1801.10095,* 2018.

[24] S. Seo, J. Huang, H. Yang and Y. Liu, "Representation learning of users and items for review rating prediction using attention-based convolutional neural network," in *3rd international workshop on machine learning methods for recommender systems (MLRec)(SDM'17)*, 2017.

[25] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, p. 2579–2605, 2008.

[26] P. Schmidt and F. Biessmann, "Quantifying Interpretability and Trust in Machine Learning Systems," *arXiv preprint arXiv:1901.08558,* 2019.

[27] M. Kusner, Y. Sun, N. Kolkin and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, 2015.

[28] G. Polčicová and P. Tiňo, "Making sense of sparse rating data in collaborative filtering via topographic organization of user preference patterns," *Neural Networks,* vol. 17, p. 1183–1199, 2004.

[29] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," *Advances in Neural Information Processing Systems,* vol. 30, p. 6754–6764, 2017.

[30] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[31] A. C. Türkmen, Y. Wang and A. J. Smola, "Fastpoint: Scalable deep point processes," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

[32] S. Liu and M. Hauskrecht, "Nonparametric regressive point processes based on conditional gaussian processes," in *Advances in Neural Information Processing Systems*, 2019.

[33] B. N. Oreshkin, D. Carpov, N. Chapados and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[34] R. Assaf and A. Schumann, "Explainable Deep Neural Networks for Multivariate Time Series Predictions.," in *IJCAI*, 2019.

[35] L. Li, J. Yan, X. Yang and Y. Jin, "Learning Interpretable Deep State Space Model for Probabilistic Time Series Forecasting.," in *IJCAI*, 2019.

[36] A. A. Ismail, M. Gunady, H. C. Bravo and S. Feizi, "Benchmarking Deep Learning Interpretability in Time Series Predictions," *arXiv preprint arXiv:2010.13924,* 2020.

[37] S. Xiao, J. Yan, X. Yang, H. Zha and S. Chu, "Modeling the intensity function of point process via recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[38] T. Hofmann, "Probmap–a probabilistic approach for mapping large document collections," *Intelligent Data Analysis,* vol. 4, p. 149–164, 2000.