UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

ANDRE PAULINO DE LIMA

Limits to surprise of recommender systems

São Paulo
2019

ANDRE PAULINO DE LIMA

Limits to surprise of recommender systems

A dissertation submitted to the School of Arts, Sciences and Humanities of the University of São Paulo in partial fulfilment of the requirements for the degree of Master of Science from the Graduate Program in Information Systems.

Concentration Area: Computing Methodology and Techniques

Revised version with the adjustments requested by the reading committee on March 15, 2019. The original version can be found in a reserved section of this Library and also in the Digital Library of Theses and Dissertations (BDTD) of this University, as required by the resolution CoPGr 6018 of October 13, 2011.

Supervisor: Professor Sarajane Marques Peres

São Paulo

2019

CATALOGUING IN PUBLICATION
(University of São Paulo. School of Arts, Sciences and Humanities. Library)
CRB 8 -4936

Lima, Andre Paulino de
Limits to surprise of recommender systems / Andre Paulino de Lima ; supervisor, Sarajane Marques Peres. – 2019
156 p. : il.

Dissertation (Master of Science) – Graduate Program in Information Systems, School of Arts, Sciences and Humanities, University of São Paulo.
Revised version.

1. Data mining. 2. Recommender systems. 3. Surprise. 4. Serendipity. I. Peres, Sarajane Marques, supervisor II. Title

CDD 22.ed.– 006.312

Dissertation authored by Andre Paulino de Lima, entitled **"Limits to surprise of recommender systems"**, submitted to the School of Arts, Sciences and Humanities of the University of São Paulo, in partial fulfilment of the requirements for the degree of Master of Science from the Graduate Program in Information Systems, in the concentration area of Computing Methodology and Techniques, approved on March 11, 2019 by the reading committee composed of the following members:

Professor Valdinei Freire da Silva

University of São Paulo

Chair of the committee

Professor Fabio Gagliardi Cozman

University of São Paulo

Professor Marcos André Gonçalves

Federal University of Minas Gerais

Professor Ivandré Paraboni

University of São Paulo

To my mother, Iraildes, who taught me how to draw sounds.

To my father, José, who taught me how to fix flip flops and give change.

To my uncle, José Augusto, who imprinted book curiosity in me.

And to my late friend and mentor, José Otávio Mariano da Silva.

Acknowledgements

First and foremost, I would like to thank my family for providing me with a nurturing, agreeable environment where I was able to invest as many hours in this project as I saw fit. In special, I thank my mother, Iraildes, and my sister, Patricia, for their trust, their patience, their encouragement, and their sense of humour, which were most needed.

I am obliged to thank my adviser, Professor Sarajane Peres, for the freedom she gave me to conduct this research project and to follow my instincts in research, for coming to my rescue at the right times and provoking the right questions, and for helping to refine my ideas and have them more clearly expressed. Sara, thank you for your trust, encouragement, and for sharing valuable bits of your professional experience with me.

I also owe thanks to Professor Helton Bíscaro and Professor Marcelo Manzato, whose comments made during the qualifying exam helped giving shape to this work. Special thanks to Professor Valdinei Silva for his support in the discussion about the approximation strategy. I also thank, without implicating, the expert teams that were consulted during this project: Professor Lúcia Barroso, Professor Antonio Lima, Piero Kauffmann, and Ana Terada, from the IME/CEA[1], for their assistance in designing the experiments; Malyina and Daniela, from the FFLCH/LLA[2], for their valuable editing support.

To my colleagues from the AI Lab team, Alexandra Diaz, Andrei Silva, Fernando Costa, and José Pagnossim, I thank for the numerous discussions on the nuts and bolts about representation of textual content, and for sharing with me their ideas and intuitions about many aspects of the relationship between representation and machine intelligence. I also thank the undergrads from the ACH2016 2017/2 class (A.I.), in which I served as an assistant in the terms of the PRPG/PAE initiative, for their engagement in the practical assignment; I have learned a lot from your doubts, and hope you have learned from mine.

The author gratefully acknowledges the financial support received from the PPgSI, which made it possible our participation in the following events: SBSI'17 and EPPC'17.

[1] Instituto de Matemática e Estatística, Centro de Estatística Aplicada.
[2] Faculdade de Filosofia, Letras e Ciências Humanas, Laboratório de Letramento Acadêmico.

"In reality, all arguments from experience are founded on the similarity which we discover among natural objects, and by which we are induced to expect effects similar to those which we have found to follow from such objects. From causes which appear similar we expect similar effects."

David Hume

(on the role of perception and similarity judgement in inductive reasoning)

"I feel passionately about measurement — about how difficult it is, about how much theory and conceptualisation is involved in measurement, and indeed, how much politics is involved."

Angus Deaton

(on the role of measurement in the collective construction of knowledge)

"A map is not the territory it represents, but, if correct, it has a similar structure to the territory, which accounts for its usefulness."

Alfred Korzybski

(on the relationship between models and reality)

"All models are wrong but some are useful."

George Box

(on the relationship between reality and models)

Abstract

LIMA, Andre Paulino de. **Limits to surprise of recommender systems**. 2019. 156 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, Brazil, 2019.

Surprise is an important component of serendipity. In this research, we address the problem of measuring the capacity of a recommender system at embedding surprise in its recommendations. We show that changes in surprise of an item owing to the growth in user experience, as well as to the increase in the number of items in the repository, are not taken into account by the current metrics and evaluation methods. As a result, in so far as the time elapsed between two measurements grows, they become increasingly incommensurable. This poses as an additional challenge in the assessment of the degree to which a recommender is exposed to unfavourable conditions, such as over-specialisation or filter bubble. We argue that a) surprise is a finite resource in any recommender system, b) there are limits to the amount of surprise that can be embedded in a recommendation, and c) these limits allow us to create a scale up in which two measurements that were taken at different moments can be directly compared. By adopting these ideas as premises, we applied the deductive method to define the concepts of maximum and minimum potential surprises and designed a surprise metric called "normalised surprise" that employs these limits. Our main contribution is an evaluation method that estimates the normalised surprise of a system. Four experiments were conducted to test the proposed metrics. The aim of the first and the second experiments was to validate the quality of the estimates of minimum and maximum potential surprise values obtained by means of a greedy algorithm. The first experiment employed a synthetic dataset to explore the limits to surprise to a user, and the second one employed the Movielens-1M to explore the limits to surprise that can be embedded in a recommendation list. The third experiment also employed the Movielens-1M dataset and was designed to investigate the effect that changes in item representation and item comparison exert on surprise. Finally, the fourth experiment compares the proposed and the current state-of-the-art evaluation method in terms of their results and execution times. The results obtained from the experiments a) confirm that the quality of the estimates of potential surprise are adequate for the purpose of evaluating normalised surprise; b) show that the item representation and comparison model that is adopted has a strong effect on surprise; and c) indicate an association between high degrees of surprise and negatively skewed pairwise distance distributions, and also indicate a significant difference in the average normalised surprise of recommendations produced by a factorisation algorithm when the surprise employs the cosine or the Euclidean distance.

Keywords: Recommender Systems. Over-specialisation. Filter Bubble. Beyond-accuracy Objectives. Surprise. Serendipity. Unexpectedness. Evaluation Method. Evaluation Metrics. Off-line Evaluation. One Plus Random. Pairwise Distance Distribution.

Resumo

LIMA, Andre Paulino de. **Limites de surpresa de Sistemas de Recomendação.** 2019. 156 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, Brasil, 2019.

A surpresa é um componente importante da serendipidade. Nesta pesquisa, abordamos o problema de medir a capacidade de um sistema de recomendação de incorporar surpresa em suas recomendações. Mostramos que as mudanças na surpresa de um item, devidas ao crescimento da experiência do usuário e ao aumento do número de itens no repositório, não são consideradas pelas métricas e métodos de avaliação atuais. Como resultado, na medida em que aumenta o tempo decorrido entre duas medições, essas se tornam cada vez mais incomensuráveis. Isso se apresenta como um desafio adicional na avaliação do grau em que um sistema de recomendação está exposto a condições desfavoráveis como superespecialização ou filtro invisível. Argumentamos que a) surpresa é um recurso finito em qualquer sistema de recomendação; b) há limites para a quantidade de surpresa que pode ser incorporada em uma recomendação; e c) esses limites nos permitem criar uma escala na qual duas medições que foram tomadas em momentos diferentes podem ser comparadas diretamente. Ao adotar essas ideias como premissas, aplicamos o método dedutivo para definir os conceitos de surpresa potencial máxima e mínima e projetar uma métrica denominada "surpresa normalizada", que emprega esses limites. Nossa principal contribuição é um método de avaliação que estima a surpresa normalizada de um sistema. Quatro experimentos foram realizados para testar as métricas propostas. O objetivo do primeiro e do segundo experimentos foi validar a qualidade das estimativas de surpresa potencial mínima e máxima obtidas por meio de um algoritmo guloso. O primeiro experimento empregou um conjunto de dados sintético para explorar os limites de surpresa para um usuário, e o segundo empregou o Movielens-1M para explorar os limites da surpresa que pode ser incorporada em uma lista de recomendações. O terceiro experimento também empregou o conjunto de dados Movielens-1M e foi desenvolvido para investigar o efeito que mudanças na representação de itens e na comparação de itens exercem sobre a surpresa. Finalmente, o quarto experimento compara os métodos de avaliação atual e proposto em termos de seus resultados e tempos de execução. Os resultados que foram obtidos dos experimentos a) confirmam que a qualidade das estimativas de surpresa potencial são adequadas para o propósito de avaliar surpresa normalizada; b) mostram que o modelo de representação e comparação de itens adotado exerce um forte efeito sobre a surpresa; e c) apontam uma associação entre graus de surpresa elevados e distribuições assimétricas negativas de distâncias, e também apontam diferenças significativas na surpresa normalizada média de recomendações produzidas por um algoritmo de fatoração quando a surpresa emprega a distância do cosseno ou a distância Euclidiana.

Palavras-chaves: Sistemas de Recomendação. Superespecialização. Filtro Invisível. Objetivos Além da Acurácia. Surpresa. Serendipidade. Método de Avaliação. Métrica. Avaliação *Off-line*. *One Plus Random*. Distribuição de Distância entre Pares.

List of figures

Figure 1 – Evolution of surprise-related metrics over time. 37
Figure 2 – Creating item vectors using a count-based DSM. 47
Figure 3 – Creating item vectors using a predictive DSM. 52
Figure 4 – Creating item vectors using a user-item matrix. 54
Figure 5 – Conceptual links between a task and method in recommendation. . . . 66
Figure 6 – Segmentation in the Netflix prize: the dataset was split into partitions M and P, and the subset of P with the highest ratings was assigned to partition T. 76
Figure 7 – A fictitious recommender system with two users and five items. 84
Figure 8 – A total sequence and an order-preserving concatenation of its slices. . . 91
Figure 9 – Description of the item vectors in the fictitious RS. 101
Figure 10 – A process to extract item vectors from the Movielens dataset. 106
Figure 11 – Histograms of pairwise distances. Each histogram shows the pairwise distance distribution obtained for the config indicated in the upper left corner. The histograms obtained from Model C are in the first row, from Model U in the second row, and from Models D and V in the third row, left and right, respectively. Histograms that are in the same column are related to the distance function indicated in the column title. Within each histogram, the dashed vertical line represents the mean of the distribution, the shaded area around it covers two standard deviations ($\mu \pm 2\sigma$) and is proportional to the coefficient of variation ($cv = \sigma/\mu$), and the triangular marker ($\curlywedge$) represents the median. Distribution skewness is indicated in the upper right corner of each histogram. Note that the ordinate is in log scale. 112
Figure 12 – Using ratings from real users in Movielens-1M to simulate test users. . 117
Figure 13 – The results obtained from the 56 configs in the experiment: the results from Model C are in the first row; from Model U in the second row; and from Models D and V in the third row, left and right, respectively. The notches around the median represent its confidence interval and were calculated using a Gaussian-based asymptotic approximation. 119

Figure 14 – Composition of the factorial experiments. The codes indicate item representations (1^{st} character, organised in rows) and distance functions (2^{nd} character, organised in columns). Incompatible combinations are marked with a "-". 120
Figure 15 – Contrasts employed to analyse effects at different resolutions of the data. In the odd groups, the results are segregated by the skewness of the distance distribution, then by intuition, and lastly by symmetry. In the even groups, the results are separated into full and factorised, and then by source of data. 120
Figure 16 – Interaction plots for contrasts applied to Groups 1, 5, and 7. Representation models in the abscissa; ordinate in normalised surprise scale. In the first column, there are the plots from the "Negatively skewed (solid line) vs other (dashed)" contrast; in the second column, the plots from the "Informational (solid) vs other (dashed)" contrast, and in the third column, the plots from the "Symmetric (solid) vs Asymmetric (dashed)" contrast. An asterisk in the caption indicates that the plot corresponds to a non-significant interaction. 123
Figure 17 – Interaction plots for contrasts applied in Groups 2, 4, 6, and 8. Distances represented in the abscissa; ordinate in normalised surprise scale. In the first column, there are the plots from the "Full (solid line) vs Factorised (dashed line)" contrast; in the second column, the plots from the "Content (solid) vs Rating (dashed) within full models" contrast, and in the third column, the plots from the "Content (solid) vs Rating (dashed) within factorised models" contrast. An asterisk in the caption indicates that the plot corresponds to a non-significant interaction. . . 125
Figure 18 – Recommended items are mapped according to the distributions employed by the recommender and the evaluator. Recommendation lists were decomposed into recommended items, which were mapped to the segment of the distribution from which they were drawn by the recommender, and also to the distance distribution employed by the evaluator to assess surprise. The counts reflect the number of recommended items that flow from the source segment to their target segment. 128

List of algorithms

Algorithm 1 – A basic recommendation method that receives a list of candidate items. . . 67
Algorithm 2 – One plus random method applied to estimate the recall of an RS 78
Algorithm 3 – One plus random method applied to estimate the surprise of an RS 79
Algorithm 4 – Off-line evaluation method to estimate surprise of an RS using $\hat{S}'_{sn}$ 97

List of tables

Table 1 – Metrics for surprise in recommender systems and their adherence to characteristics presumed by the two cognitive models. A metric is adherent to the notion of subjectivity if it considers that surprise not only depends on the user experience (column "is subjective"), but also that this experience depends only on information that is endogenous to the user (column "is intrinsic"). 44

Table 2 – Combinations of item representation and item comparison explored by the recommendation algorithms. They gradually move from adopting item representation and comparison schemes that are independent of those used to assess surprise (FP) towards adopting the same item representation and comparison that are used by the surprise metric (N3). 69

Table 3 – Greedy approximations to potential surprise in an RS with a synthetic repository. Values of D_{max} or D_{min} represent the relative difference between the exact and the approximate potential surprise. Values under 0.01% are written as a dash. 102

Table 4 – Attributes of the extended MovieLens-1M dataset. The dataset contains 6,040 users, 3,883 items, 1,000,209 ratings and was extended with 3,804 short textual descriptions of the items. Each user has at least 20 ratings. 104

Table 5 – Examplars within each segment of the token per document distribution. 110

Table 6 – Greedy approximations to potential surprise in a sample of the Movielens-1M. Values in the *max* or *min* columns are the relative difference between exact and approximate values of potential surprise. Dashes represent values under 0.01%. 115

Table 7 – Effect sizes obtained from linear regression models fit to the relative difference between exact and approximate values of potential surprise. In the column "Model", the model equation is defined: the factors *repmodel*, *distfn*, and *length* respectively correspond to item representation, distance function, and length of the sequence. R^2 represents the amount of variation that can be explained by the model (systematic variation). . 115

Table 8 – Mean normalised surprise obtained from the 56 configs in the experiment. Columns “mean” and “ci” (confidence interval) report on the results obtained from the recommendation algorithm indicated in the major column. Rows are listed in increasing order of skew and coefficient of variation (cv), and grouped as negatively skewed, fairly balanced ($|skew| < 0.5$), and positively skewed. Entries filled with a dash correspond to incompatible configurations. Intervals estimated at a 95% confidence level, with margin of error being inferior to 5%. 118

Table 9 – Significance and effects sizes of the contrasts applied to the results from each group. Results were considered significant at $p < 0.05$. Effect sizes were reported in the r scale. In the last column, the direction of change is described as moving from the right to the left condition of the contrast. For example, in the “Negatively-skewed vs other” contrast in Group 1 (first row in the table), the increase in mean normalised surprise in Model C occurs if one moves from the “other” condition to the “Negatively-skewed” condition. 122

Table 10 – Summary of patterns and discrepancies found in the analysis. In the column “Pattern”, it is described a pattern that is salient or expected at a given level of the results. For example, in the first row, $\bar{S}_{sn}|_{N_i} \equiv \bar{S}_{sn}|_{N_j}$ means that the mean normalised surprise values obtained from any two neighbourhood-based algorithms, N_i and N_j, are predominantly indistinguishable from one another. In the columns “Scope”, “Grp” (group), and “Lev” (level), it is indicated the level of the results in which the pattern holds. The exceptional instances are described in the last column. 126

Table 11 – Comparison of the results obtained from the one plus random and the proposed method applied to our extended Movielens 1M dataset. Algorithms indicated as MF and FP are factorisation-based algorithms and correspond to the PureSVD, with $f = 50$ dimensions, whereas IB and N2 are neighbourhood-based algorithms, with $k = 50$. Values reported as averages, except Time (total, in seconds), recall and surprise (15% confidence interval at 95% confidence level). 136

List of abbreviations and acronyms

BMT	Bayesian Multiplicative Treatment
CBRS	Content-Based Recommender System
CFRS	Collaborative Filtering Recommender System
CoDA	Compositional Data Analysis
DRS	Demographic Recommender System
DSM	Distributional Semantic Model
IMDb	Internet Movie Database
KBRS	Knowledge-Based Recommender System
kNN	k-Neareast Neighbour algorithm
MAE	Mean Absolute Error
MLE	Maximum Likelihood Estimation
PPM	Primitive Prediction Model
RMSE	Root Mean Square Error
RQ	Research Question
RS	Recommender System
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
tf-idf	term frequency-inverse document frequency (weighting scheme)

List of symbols

x	A generic variable (lower case, regular font)
X	A generic set (upper case, regular font)
$\mathbf{X}$	A generic matrix (upper case, bold font), with elements x_{ij}
$\mathbf{X}_i$	The i^{th} row (or column) vector from the matrix $\mathbf{X}$
$\mathbf{x}$	A vector (lower case, bold font), or item vector, with elements x_i
$\|X\|, \|seq\|$	The number of elements in the set X or in the sequence seq, respectively
$\|\|\mathbf{x}\|\|_k$	The L_k norm of the vector $\mathbf{x}$ (default is $k = 2$)
I	The set of items in the repository of a recommender system
U	The set of users of a recommender system
E_u	The set of items to which the user u has been exposed
N_u	The set of items unknown to user u
seq, $tseq$	A sequence and a total sequence of items, respectively
L	A recommendation list (corresponds to a sequence of items)
$\mathbf{D}$	The document-term matrix, with elements d_{it}, which is the number of occurrences of the term t in the document combined with the item i
$\mathbf{R}$	The user-item matrix, or rating matrix, with elements r_{ui}
$r_{ui}, \hat{r}_{ui}$	The actual and the predicted rating given by user u to item i, respectively
$\mathbf{V}$	The item vector matrix, with $\mathbf{V}_i$ as the i^{th} row of the matrix $\mathbf{V}$
$\mathbf{M}, \boldsymbol{\rho}, \mathbf{S}$	The distance, similarity, and weighted similarity matrices, respectively
$\mathbf{N}$	A count matrix, with n_{ij} as the number of users that rated items i and j
$\mathbf{B}$	The baseline predictors matrix, with b_{ui} as the expected rating $\hat{r}_{ui}$
$\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{Q}$	The matrices resulting from applying truncated SVD to the $\mathbf{R}$ matrix.

$S_i(i, E_u)$ surprise of the item i to the user u, or given the user experience E_u

$S_s(seq, E_u)$ surprise of the sequence seq to the user u, or given the user experience E_u

$S_{pmax}(N_u, E_u)$ maximum potential surprise of N_u given E_u,
or the maximum amount of surprise that the system can offer to user u

$S_{pmin}(N_u, E_u)$ minimum potential surprise of N_u given E_u
or the minimum amount of surprise that the system can offer to user u

$S_{sn}(tseq, E_u)$ normalised surprise of the (total) sequence $tseq$ given E_u
or the normalised surprise embedded in the sequence $tseq$

$\hat{S}_{pmax}(N_u, E_u)$ estimated maximum potential surprise of N_u given E_u

$\hat{S}_{pmin}(N_u, E_u)$ estimated minimum potential surprise of N_u given E_u

$\hat{S}_{sn}(tseq, E_u)$ estimated normalised surprise of the (total) sequence $tseq$ given E_u

$S'_{pmax}(N_u, E_u, k)$ maximum potential surprise of a sequence of length k given E_u,
or the maximum amount of surprise to the user u that can be embedded
in a sequence of length k composed of items in N_u

$S'_{pmin}(N_u, E_u, k)$ minimum potential surprise of a sequence of length k given E_u
or the minimum amount of surprise to the user u that can be embedded
in a sequence of length k composed of items in N_u

$S'_{sn}(seq, E_u, k)$ normalised surprise of the (partial) sequence seq given E_u
or the normalised surprise embedded in the sequence seq

$\hat{S}'_{pmax}(N_u, E_u, k)$ estimated maximum potential surprise of a sequence of length k
from N_u given E_u

$\hat{S}'_{pmin}(N_u, E_u, k)$ estimated minimum potential surprise of a sequence of length k
from N_u given E_u

$\hat{S}'_{sn}(seq, E_u, k)$ estimated normalised surprise of the (partial) sequence seq given E_u

Contents

1 Introduction 20

1.1 Objectives 25

1.2 Research methods 26

1.3 Contributions 28

1.4 Dissertation outline 29

2 Background 31

2.1 A taxonomy of recommender systems 31

2.2 The property of surprise 35

2.3 The metrics for surprise 36

2.3.1 A metric for unexpectedness 38

2.3.2 A metric for serendipity 38

2.3.3 A metric for general unexpectedness 39

2.3.4 A second metric for unexpectedness 39

2.3.5 The unserendipity metric 40

2.3.6 A metric for surprise 40

2.3.7 Summary and analysis 41

2.4 The experience of surprise 43

2.5 The item representation models (or how item vectors are extracted) . 45

2.5.1 Item representation in CBRS 46

2.5.2 Item representation in CFRS 53

2.5.3 Summary and analysis 55

2.6 The item comparison functions (or how item vectors are compared) . 56

2.6.1 Metric, distance and similarity 56

2.6.2 Distance functions based on geometric intuition 59

2.6.3 Distances based on combinatorial intuition 60

2.6.4 Distances based on informational intuition 61

2.6.5 Distances based on compositional intuition 63

2.6.6 Summary and analysis 64

2.7 The recommendation algorithms 65

2.7.1 Recommendation methods . . . 65
2.7.2 Recommendation algorithms . . . 68
2.8 The evaluation methods . . . 72
2.8.1 A brief taxonomy of experiment designs . . . 73
2.8.2 Off-line evaluation methods . . . 75
2.9 Closing remarks . . . 80

3 Foreground . . . 81
3.1 Revisiting the property of surprise . . . 81
3.2 The behaviour of the S_i metric . . . 84
3.3 A theoretical model of potential surprise . . . 85
3.3.1 Surprise as a finite resource . . . 87
3.3.2 The potential surprise . . . 89
3.3.3 The normalised surprise of a total sequence . . . 91
3.3.4 Computational costs and approximations . . . 92
3.3.5 The normalised surprise of a recommendation list . . . 93
3.4 Revisiting a premise of the model of potential surprise . . . 94
3.5 Adapted evaluation method . . . 96
3.6 Closing remarks . . . 98

4 Experiments . . . 99
4.1 Experiment 1: evaluating approximations to the potential surprise of a system . . . 100
4.2 Experiment 2: exploring the Movielens dataset . . . 102
4.2.1 Extending the Movielens dataset . . . 103
4.2.2 Preprocessing the extended Movielens dataset . . . 104
4.2.3 Analysing results from the Common Steps . . . 109
4.2.4 Analysing results from the Final Step . . . 110
4.2.5 Evaluating approximations to the potential surprise of a sequence . . . 113
4.2.6 Summary of the results . . . 114
4.3 Experiment 3: analysing change in surprise owing to a change in config 116
4.3.1 Experiment design . . . 116
4.3.2 Experiment results . . . 118

4.3.3 Analysis of the results . . . 119
4.3.4 Hypotheses to integrate the found patterns and discrepancies . . . 126
4.4 Experiment 4: comparing current and proposed evaluation methods . . 133
4.5 Discussion of the evidences . . . 137

5 Conclusion . . . 138
5.1 Limitations and risks . . . 139
5.2 Opportunities for improvement and alternative directions . . . 140
5.3 Final remarks . . . 142

Bibliography . . . 143

APPENDIX A – Formal definition of the proposed metrics . 151

APPENDIX B – Sample size for a population mean . . . 154

APPENDIX C – Doc2Vec parameters employed in Model D 156

1 Introduction

A few decades have passed since the recommender systems (RS) first emerged. They now exert a ubiquitous influence on private and public affairs around the world. This influence results from their application to domains that range from entertainment to news, and from social networks to business and professional spheres. They usually serve as a component in a larger system or platform, and their purpose is straightforward: a recommender system must enable its user to select, or to be exposed to, something of interest within a universe of items that is largely unknown to them. In principle, items may represent anything of interest to the users, such as music, movies, books, hotels, restaurants, news articles, or scientific literature. In social networks, recommender systems highlight posts, send messages, suggest connections to new friends, and seek dating partners. In other words, recommender systems act largely as information filters within larger systems.

However, as with the introduction of any new technology, they provide benefits but also pose risks. In particular, the popularisation of social networks, combined with the dismantling and digitisation of the traditional media institutions, has paved the way to a global shift in news consumption habits. This situation has exposed an inherent weakness of the recommender systems in social networks: its information filtering capacity can be exploited for political strategies. Reports have recently come to the public attention, describing formally organised social media manipulation campaigns in 48 countries, including Brazil, the UK, and US (BRADSHAW; HOWARD, 2018). In fact, even without the interference of external actors, a recommender may involve its user in a "self-reinforcing pattern of narrowing exposure" to items (NGUYEN et al., 2014), a condition referred to as over-specialisation, which is known to affect some categories of recommenders.

The risk of adopting technologies that can foster large scale opinion forming and social polarisation has been long feared by political scientists and journalists. The term "filter bubble", coined by Pariser (2011), echoes ideas that date back to the warnings about the potential loss of serendipity[1] in information filtering technologies that were issued by Gup (1997). Both endogenous (over-specialisation) and exogenous (external interference) factors contribute to this phenomenon, but it is difficult to operationalize the notions of filter bubble, over-specialisation, and external interference, and assessing the degree

[1] For the purpose of this study, the term "serendipity" refers to the discovery of valuable or pleasant things which one had not actively sought.

to which a recommender system is exposed to these conditions is still an open question. However, there seems to be a consensus among researchers with regard to the way these conditions can be attributed to a systematic reduction of serendipity in personalised recommendations. For this reason, in the following section we clarify the links between personalised recommendations, personalisation approaches, and serendipity.

The link between a personalisation approach and beyond-accuracy objectives

In its most rudimentary form, a recommender system relies solely on descriptive statistics to make non-personalised recommendations, which comprises items of popular interest. However, personalised recommendations have a higher added value, which is explained not only by the benefits it brings to the user but also by the benefits it brings to the owner of the system, who can encourage the exploitation of items that are less known to the users. In this regard, numerous approaches have been devised to enable recommender systems to produce personalised recommendations. In seeking to achieve its goal, a personalisation approach explores diverse sources of information, such as the features of the items, the demographic characteristics of the users, the traces of user activity in the platform, or the network of social relationships that interconnects the users of the system (BURKE, 2007).

Since several personalisation approaches are available, to determine the most appropriate one for a given domain (e.g., movie or music recommendation), it is necessary to select (or devise) an evaluation method that can measure the performance of each candidate personalisation approach, and determine which one makes the underlying system more competent in that domain. This evaluation method must take account of the following: a) the **objective**(s) driven by the application domain; b) the system **property**(ies) that can assist in achieving the objective(s); and c) the **metric**(s) that can be used to assess the degree to which the contributory property(ies) is(are) present in the system.

The usual drive of recommender systems in the initial studies in the literature was to recommend the most promising item that was unknown to the user (CASTELLS; HURLEY; VARGAS, 2015). In such a setting, a) the objective was to predict the rating a user would give to an item as accurately as possible; b) the prediction accuracy was the only relevant property; c) metrics like RMSE (Root Mean Square Error) or MAE

(Mean Absolute Error) were used to estimate the prediction error (and were aggregated to express a system-level property). It must be noted that the concept of "objective" is different from that of a system property, because the former represents a combination of a property with an optimisation directive. For example, in the objective "accuracy must be maximised", accuracy is the system property, and minimisation is the directive.

However, as the research area evolved, a consensus emerged that it was not enough to adopt accuracy as the only objective to be pursued (HERLOCKER et al., 2004; MCNEE; RIEDL; KONSTAN, 2006). Many recent studies in the literature on recommender systems have highlighted other desirable objectives, such as coverage, diversity, serendipity and surprise; these are the so-called "beyond-accuracy objectives" (GE; DELGADO-BATTENFELD; JANNACH, 2010; KAMINSKAS; BRIDGE, 2016). In this new context, an objective is a more holistic notion of user satisfaction, which is combined with a set of contributory properties, in contrast with the idea of regarding accuracy as the single contributory property.

The property of surprise is of interest to this project owing to its link to serendipity: according to Herlocker, Konstan and Riedl (2002), serendipity results from the interaction between surprise and relevance. Although it has been much less investigated than relevance, surprise has been the focus of many works, and there is a degree of consensus that surprise can be defined in terms of a recommendation that was unexpected, and is unfamiliar to the user (MURAKAMI; MORI; ORIHARA, 2008; GE; DELGADO-BATTENFELD; JANNACH, 2010; AKIYAMA; OBARA; TANIZAKI, 2010; ADAMOPOULOS; TUZHILIN, 2011; ZHANG et al., 2012; KAMINSKAS; BRIDGE, 2014). The literature on surprise in cognitive science offers another useful perspective: quantitative models of surprise portray surprise as a cognitive emotion that emerges when the subject (user) is faced with new information (REISENZEIN; HORSTMANN; SCHÜTZWOHL, 2017; BARTO; MIROLLI; BALDASSARRE, 2013; ITTI; BALDI, 2009). From this perspective, a recommendation can be framed as a vehicle that conveys information to the user, and the surprise embedded in the recommendation corresponds to the amount of new information that is retained by the user.

The links between surprise and over-specialisation

As mentioned earlier, over-specialisation and filter bubble are conditions that arise from a systematic reduction of serendipity in personalised recommendations. We argue

that the surprise component of serendipity has a higher association with these conditions than has relevance because practical personalisation approaches include relevance as the critical property. For example, suppose there are two recommender instances A and B, with A operating in a normal condition, B operating in an over-specialised condition, and both instances were optimised to produce highly relevant recommendations. In this scenario, recommendations from instance A will be, on average, less familiar to the user than those from B. The degree to which a recommender system is exposed to an over-specialised or a filter-bubble condition can be assessed as a function of how much surprise a recommender system has embedded in its recommendations at two different points in time, for example:

$\Delta S = S(t+1) - S(t)$, where $S(t)$ is an estimate of the system-level surprise at time t.

The challenge of detecting this change (ΔS) is that the system-level surprise relies on the item-level surprise, which changes in time as a response to ordinary events in the life-cycle of a recommender system, such as the user being exposed to new items (i.e., the more a user "knows" about the domain, the less surprise they should experience), or the introduction of new items in the repository (i.e., the more there is "to know", the more surprise the users should experience). The current metrics for surprise seek to estimate the surprise of a user, and some of them explicitly account for the growth in user experience. However, as they are based on a user-centric perspective, none of them account for the increase in the number of items in the repository. This deficit makes it difficult to detect variation in system-level surprise. To make this point more clear, first we explore a scenario in which the variation in surprise for a single user is estimated, and then show how a system-level estimate fails to account for the introduction of new items in the repository:

Scenario 1.1 *Estimating the variation of surprise for a single user.*
Preliminaries:

1. *Let I represent the set of items in the repository of the recommender system.*
2. *Let E_u, the set of items known to the user u, represent the user experience.*
3. *Let $s(L, u)$ be a metric that estimates the surprise that the user u would experience from being exposed to the items in the recommendation list L. This metric produces values in the real interval $[0, N]$, with $N = |L|$. If $s(L, u) = 0$ then the items in L are absolutely familiar to the user u, so no surprise would be experienced by them. In contrast, if $s(L, u) = N$ then L embeds the highest possible surprise to u.*

Initial conditions:

4. *At time t, the user experience is $E_u(t)$.*
5. *At time t, the set of items in the repository is $I(t)$.*

Interactions:

6. *An estimate of user surprise is obtained at time t: $m(t,u) = s(L_u(t), u)$; L_u is a recommendation list produced by the system to the user u for an estimation purpose (i.e., the list is used by the estimation process, but it is not presented to the user).*
7. *A non-empty set of new items, $\Delta I \mid I \cap \Delta I = \varnothing$, is introduced in the repository.*
8. *An estimate of user surprise is obtained at time $t+1$: $m(t+1,u) = s(L_u(t+1), u)$.*

Final conditions:

- *The user has not been exposed to new items, thus $E_u(t) = E_u(t+1)$.*
- *The set of items has increased in number of items, $I(t+1) = I(t) \cup \Delta I$.*
- *The estimated variation in the amount of surprise that was embedded in the recommendation lists produced to the user u is $\Delta s(u) = m(t+1,u) - m(t,u)$.* ■

In summary, this scenario describes a situation in which the user was not exposed to new items, but there was an increase in the number of items in the repository. Now suppose that, instead of obtaining the estimated variation in surprise for a single user, we obtained the estimated variation in surprise for a sample of n users, $U' = \{u_1, u_2, \ldots, u_n\}$. Assume that no variation was detected for any user, i.e. $\Delta s(u) = 0, \forall u \in U'$, and that a massive increase in the number of items in the repository occurred, with $|\Delta I| > |I| \times 2$. Finally, let $S(t) = \frac{1}{n} \sum_{u \in U'} m(t,u)$ be a metric that estimates the system-level surprise at time t by averaging the estimates of user surprise. Thus, no variation in system-level surprise is detected, because:

$$\Delta S = S(t+1) - S(t) = \frac{1}{n} \sum_{u \in U'} m(t+1,u) - \frac{1}{n} \sum_{u \in U'} m(t,u)$$
$$= \frac{1}{n} \sum_{u \in U'} m(t+1,u) - m(t,u) = \frac{1}{n} \sum_{u \in U'} \Delta s(u) = 0.$$

As will be seen in Chapter 3, we argue that surprise is a limited resource in any recommender system, and that the increase in the number of items in the repository implies that a larger amount of surprise is now available to each user. Taking this into

consideration, let $s_{max}(t, u)$ represent the maximum amount of surprise that was available to the user u at time t. Then, it is the case that $s_{max}(t+1, u) > s_{max}(t, u)$. The fraction of the total amount of surprise that was embedded in the recommendation list $L_u(t+1)$, namely $s(L_u(t+1), u)/s_{max}(t+1, u)$, is smaller than the fraction of the total amount of surprise that was embedded in $L_u(t)$, namely $s(L_u(t), u)/s_{max}(t, u)$, because if we take v such as $v = s(L_u(t+1), u) = s(L_u(t), u) \geq 0$, we have that:

$$s_{max}(t+1, u) > s_{max}(t, u) \implies \frac{v}{s_{max}(t+1, u)} \leq \frac{v}{s_{max}(t, u)}.$$

This inequality indicates that a reduction in the system capacity to embed surprise in its recommendations took place, but the change was not detected using the $S(t)$ metric because it leaves the inflation of the set of items uncovered. As a result, in so far as the time elapsed between two measurements grows and the size of the repository increases, the measurements become gradually incommensurable.

1.1 Objectives

To fill in this gap, we pursue a metric for surprise that makes measurements taken at two points in time directly comparable. We argue that framing the problem of measuring surprise as the ratio between the actual and the potential system performance (i.e., between "the amount of surprise that the system is currently embedding in its recommendations" and "the maximum amount of surprise that the system can embed in its recommendations at this point in time") confines the effects of the aforementioned changes to a measurement. This led us to the following research questions, which inquires into the limits to surprise of a recommender system and considers the life-cycle events that change surprise from both user and systemic perspectives:

RQ1 Are there limits to how much surprise a recommender can offer to a user?

RQ2 Are there limits to how much surprise can be embedded in a recommendation?

RQ3 If these limits exist, is it possible to use them to create a measurement scale up in which the performance of a system can be measured?

RQ4 If these limits exist, are they influenced by decisions on how to represent items or which distance function to employ to compare these item representations?

The primary objective of this research is to develop a metric with the properties described earlier, and our main contribution is an evaluation method that factors the growth in user experience and the growth in the size of the repository into the estimate of the amount of surprise a recommender is embedding in its recommendations. To fulfil the primary objective, several new concepts are defined on the basis of a "theoretical model of potential surprise", and a metric is created for each of these concepts; the implementation and verification of these metrics comprise the secondary objectives of this research.

It must be noted that the research questions do not aim to determine if the current intuitions on surprise correspond to the surprise experienced by real users when interacting with a system, nor do they aim to assess effect that changes in the parameters of a recommender system exert on user experience of surprise. Instead, they seek to investigate the effect on system-level surprise of distinct, but plausible, models of surprise, represented by different forms of the item-level surprise metric, as will be clarified in Chapter 3.

1.2 Research methods

To approach the research questions, this project employs two methods. We tackle the questions RQ1 to RQ3 by applying the deductive method on premises collected from the literature on recommender systems (specifically on surprise metrics) and cognitive science (specifically on quantitative models of surprise). The fourth question is approached with an empirical, quantitative method, which it is based on collecting and analysing data produced in a computational simulation designed to evaluate RQ4 (Experiment 3). However, the latter method is also used to assess the soundness of a premise adopted to tackle RQ1 to RQ3: that adequate estimates to potential surprise can be obtained by adopting a greedy approximation strategy (Experiments 1 and 2). In addition, the empirical method is also used to compare the execution times and results obtained from the proposed and the current state-of-the-art evaluation methods (Experiment 4). The objectives and major details of each experiment are described in the following paragraphs.

Experiment 1: evaluating approximations to the potential surprise of a system

As will be described in Chapter 3, the proposed model for potential surprise defines the concepts of maximum potential surprise and minimum potential surprise of a system. However, these quantities can only be approximated in realistic settings. The aim of this experiment was to assess the quality of approximations to maximum and minimum potential surprise obtained by a greedy approximation strategy. This experiment employed a small synthetic dataset to assess the differences between exact and approximate computations of maximum and minimum potential surprises. Since the definition of surprise is based on the notion of distance, several distance functions were explored, covering a range of intuitions on how to measure the separation or difference between two objects: non-normalised Euclidean distance, cosine distance, Jaccard distance, Kullback-Leibler divergence, Jensen-Shannon divergence, and Aitchison distance.

Experiment 2: exploring the Movielens dataset

The purpose of this experiment was to explore the Movielens dataset (HARPER; KONSTAN, 2015) by a) looking for possible limitations to its use in this project (e.g. missing attributes, entity without required relationship); b) evaluating the quality of the short textual movie descriptions that were combined with the original dataset (e.g. checking for missing description, descriptions that are too short, or descriptions in a language different than English), and c) characterising the data distribution according to different data representation models (e.g. distributional semantic models, user-item vector models) and distance functions. The textual descriptions were combined with the dataset to allow the exploration of content-based personalisation approaches. Moreover, an experiment similar to the experiment 1 was performed: the synthetic dataset was replaced by data sampled from the Movielens dataset, and the aim was to assess the quality of approximations to maximum and minimum potential surprises of a sequence.

Experiment 3: analysing change in surprise owing to a change in config

The aim of this experiment was to assess the behaviour of the proposed theoretical model for surprise under several experimental conditions that varied the item representation model and the distance functions that were employed in the definition of the surprise of an item. To conduct this experiment, a controlled experimental environment was developed. Four data representation models were explored: a) the traditional vector space model of semantics (or count-based distributional semantic model), b) a distributed vector space model of semantics (or predictive distributional semantic model), c) the user-item representation model, and d) a factorised model based on the user-item matrix, which is usually employed in collaborative recommenders. In addition, six distance functions were explored: a) the non-normalised Euclidean distance, b) the cosine distance, c) the Jaccard distance, d) the Kullback-Leibler divergence, e) the Jensen-Shannon divergence, and f) the Aitchison distance. Moreover, three recommendation algorithms were explored: the PureSVD and three variations of the item-kNN algorithm.

Experiment 4: comparing current and proposed evaluation methods

The objective of this experiment was to compare the current state-of-the-art evaluation method for surprise (the one-plus-random method) to the proposed evaluation method, reporting difference in their results, as well as in their average execution times. To achieve this purpose, we replicated part of the original study (KAMINSKAS; BRIDGE, 2014), which employed the Movielens-1M dataset to build the repository of the recommender, and the Jaccard distance in the definition of surprise of an item.

1.3 Contributions

The main contribution of this dissertation is a method to detect if an instance of recommender system is in an overspecialisation or filter bubble condition. This method can assess the degree to which the personalisation components of social media platforms or search engines are fostering a systematic reduction of the universe of information being

offered to the user, and can be applied to any recommender systems in which items are represented as vectors (e.g. collaborative or content filtering recommenders).

A second contribution is the finding of a relationship between the distance function employed in the model of surprise and the estimate of the capacity of a recommender at embedding the available surprise into new recommendations. This finding adds a qualified argument to the discussion on the need of validated models of surprise in recommender systems because the choice of a distance function can lead to very different estimates depending on the codomain of the distance (e.g. bounded or unbounded function).

Regarding validated models of surprise, there was a recent call for researchers in this area to seek consilience with other research areas, such as psychology, to improve the capacity of fundamental models to properly predict human responses when interacting with recommender systems (FERRO et al., 2018). In this study, we took some ideas from the cognitive sciences about how human beings experience surprise to tackle with the lack of validated models of surprise in the literature on recommender systems. In this effort, we sought for common essential aspects of two competing models of surprise (that have some empirical support) from the literature on cognitive sciences to select a metric for surprise of recommender systems. We also identified how different models of preference and surprise, which aim to predict user responses when exposed to new items, can be framed as operations over representations, a perspective that is usually explored to produce architectural level descriptions in computational cognitive science (DAWSON, 2013).

1.4 Dissertation outline

The remainder of this text is organised as follows:

- In Chapter 2, the theoretical foundation necessary to understand the problem approached by this work is presented. The chapter begins with the description of a taxonomy of recommender systems that is based on personalisation approach (Section 2.1). We refer back to this taxonomy in several sections, which reflects its role in integrating different concepts presented along the text. It is followed by a critical review of the metrics for surprise in the literature (Sections 2.2, 2.3, and 2.4). In Section 2.5, four item representation models that can be used to encode items as item vectors are presented, and six distance functions that can be used to

measure (dis)similarity between item vectors are described in Section 2.6. The main recommendation algorithms employed in content-based and collaborative approaches are described in Section 2.7. A discussion on the current state-of-the-art evaluation method for surprise is presented in Section 2.8. The chapter ends with a concise description of how surprise metrics and recommendation algorithms relate to the concepts of item representation and item comparison.

- In Chapter 3, the first three research questions are addressed. It begins with a discussion on the properties that a metric for surprise should have (Section 3.1). The discussion adopts some ideas that were described in the critical review as premises, and concludes by justifying the selection of the surprise metric that is in higher agreement to the desired properties. In Section 3.2, the behaviour of the selected metric is analysed, and its fundamental ideas that will be integrated in the theoretical model are highlighted. The theoretical model developed in Section 3.3 is employed to estimate the maximum and minimum amounts of surprise that a recommender can offer to a user, and to create an evaluation method for surprise (Section 3.5). The chapter closes with a summary of how the theoretical model answers RQ1 to RQ3.

- In Chapter 4, the last research question is addressed, as well as some premises assumed by the theoretical model. In Section 4.3, the results from an experiment designed to address RQ4 are reported, while Sections 4.1 and 4.2 are predominantly dedicated to assess the quality of greedy approximations to potential surprise. The experiment reported in Section 4.4 compares results from the current and proposed evaluation methods for surprise. The chapter ends with a discussion on how the results obtained support the validity of the theoretical model.

- Finally, in Chapter 5 concludes this work by presenting known limitations of this work and risks to the validity of this research, as well as directions for future work.

2 Background

In this chapter, we review the literature related to the problem of measuring surprise in recommender systems. In Section 2.1, a concise taxonomy is presented, in which a recommender is classified according to the personalisation approach it adopts. The notion of surprise in recommender systems is discussed in Section 2.2, and a critical review of the metrics for surprise in the literature is presented in Sections 2.3 and 2.4. As will be seen, the notion of surprise is closely related to the idea of measuring dissimilarity between mathematical objects, such as sets or vectors. For this reason, we also review a number of representation models that are commonly used to create item vectors (Section 2.5), as well as some distance functions that can be used to compare such vectors (Section 2.6). The concepts of item representation and item comparison are integrated in Section 2.7 to explain the inner workings of recommendation algorithms, in particular the ones that follow the neighbourhood-based or the factorisation-based approach. Finally, the evaluation methods used to obtain system-level estimates of surprise are analysed in Section 2.8.

2.1 A taxonomy of recommender systems

As mentioned earlier, numerous approaches have been proposed over time to enable an RS to produce personalised recommendations. Each approach explores different sources of information, such as the features of the items in the repository, the characteristics of the users of the system, the history of user interactions with the RS or the network of social relationships that connects the users. Burke (2007) provides a taxonomy that is usually cited in works in the area because it systematically groups RS according to the personalisation approach they adopt. This taxonomy can be made more clear if we describe its categories in terms of how an RS responds to recurring events in its life cycle, such as:

a) When producing a new recommendation, what information is employed?
b) When a new item is inserted in the repository, and thus it does not have any ratings, can the system recommend it?
c) When a new user subscribes to the RS, and therefore they have not given any ratings, can the system recommend any items to them?

d) As the number of ratings of a given user increases, can one expect any change in the quality of the recommendations produced to them?

In addition, as will be required in the next section, it is important to determine if a recommender system keeps numerical representation of its items. This is important not only because recommendations methods usually need to assess the (dis)similarity between items in a regular basis, but also because numerical representations are required to assess surprise of a recommendation list. With these characteristics in mind, in following we describe the Burke's taxonomy of RS:

- Content-Based Recommender Systems (CBRS) employ the ratings that were given by the users to the items, as well as features of the items in the repository, to produce new recommendations (GEMMIS et al., 2015). This category assumes that each item has some content combined with it, and that such content can be used to assess its similarity to other items in the repository. A recommendation will consist of new items that are similar to the items to which the user has given high ratings. In fact, a key success factor for CBRS is the quality of the content combined with each item: items that are similar, according to the user judgment, must have similar attributes according to the representation model adopted. In addition, as will be described in Section 2.5, if items are combined with a textual content, vector space models of semantics (TURNEY; PANTEL, 2010; BARONI; DINU; KRUSZEWSKI, 2014) can be used to produce numerical representations for these items. With regard to life cycle events, CBRS can be described as follows: a) a new recommendation to user u exclusively depends on their past ratings and the content of the rated items; b) an item that has not been rated by any user can be promptly recommended to users; c) recommending items to a user that have not rated any item is expected to be unsuccessful; and d) as the number of ratings given by a user increases, new recommendations tend to be similar to items highly rated by the user in the past; in the filter bubble scenario (NGUYEN et al., 2014), new items will be too similar to items known to the user, which is a limitation also referred to as the over-specialisation problem.

- Collaborative Filtering Recommender Systems (CFRS) exclusively employ ratings to produce new recommendations (NING; DESROSIERS; KARYPIS, 2015; KOREN; BELL, 2015). A key idea in these systems is to explore the similarity between users (or

items) to produce new recommendations. For example, suppose that two users, u_1 and u_2, have given the same rating to 100 items. In addition, suppose that the user u_1 has rated one extra item, namely i_{u_1}, that user u_2 has not rated. A CFRS explores the fact that users u_1 and u_2 have similar preferences for items to predict that user u_2 will give to the item i_{u_1} the same rating attributed by user u_1 to that item. Recommenders in this category usually organise all the ratings in a user-item matrix $\mathbf{R}$[1], where each element, r_{ui}, represents the rating given by user u to item i. This matrix can produce a numerical representation of a user (a row-vector of $\mathbf{R}$) or of an item (a column-vector of $\mathbf{R}$), and different recommendation methods were devised to explore these numerical representations CFRS are less prone to over-specialisation. For example, user-based recommendation methods explore the similarity between user vectors to produce a new recommendation, as described in the previous example. In contrast, item-based recommendation methods explore the similarity between item vectors to produce new recommendations. The number of items rated by any user is expected to be a small if compared to the total number of items, so the matrix $\mathbf{R}$ must be sparse; different solutions have been conceived to deal with sparsity, like neighbourhood-based and factorisation-based methods. Finally, the four salient features of this category are a) a new recommendation to user u can use the whole $\mathbf{R}$ matrix; b) an item that has not been rated by any user cannot be promptly recommended to users; c) recommending items to a user that have not rated any item is expected to fail; and d) they are less prone to over-specialisation because new recommendations do not rely solely on user past ratings.

- Knowledge-Based Recommender Systems (KBRS) combine information about users and items with specific domain knowledge to produce recommendations (FELFERNIG et al., 2015). User information is represented as requirements explicitly presented by the user for an eligible recommendation, which are employed to select an item. In turn, an item is represented by the attributes that are usually considered by the user when selecting an item. In addition, domain knowledge is represented as heuristics collected from domain specialists, which are incorporated into the system to drive an interactive recommendation process. This category of RS is commonly

[1] More commonly referred to as the ratings matrix, but we prefer user-item matrix because it makes more clear how dimensions are organised in the matrix.

adopted in domains where the frequency of item consumption is low. In such domains, approaches like the ones employed by CBRS and CFRS are expected to fail owing to severe sparsity effects. Two significant sub-categories of KBRS are a) Case-Based Recommender Systems, in which items (or cases) are usually handled as multidimensional vectors (each dimension representing a case component) (LORENZI; RICCI, 2005), and b) Constraint-Based Recommender Systems, which handle items as collections of attributes and values and, thus, do not require numerical representations (FELFERNIG et al., 2015).

- Demographic Recommender Systems (DRS) are systems that explore demographical characteristics of their users to produce recommendations, such as age, gender, place of residence, education, marital status, or current employment status (BELIAKOV; CALVO; JAMES, 2015; PAZZANI, 1999). Each user is assigned to one or more demographic classes (or user stereotypes) and, thus, a primary success factor for this category of recommender systems is effectively identifying groups of users with similar preferences. Given that a user may have different degrees of membership to each demographic class, a vector representation can be devised to produce a numerical representation for the preference of a user. Following the same idea, a vector representation for items can be devised as the expected rating that individuals in the demographic classes would attribute to the item.

- Hybrid Recommender Systems combine different approaches to overcome limitations associated to each individual approach. For example, CFRS are not effective in recommending unrated items. This limitation, known as the "cold-start problem", does not occur in CBRS, because new recommendations are produced by measuring similarity between the content of items. In contrast, CBRS tend to produce recommendations biased by the items highly rated by a user in the past (over-specialisation problem), whereas CFRS are less prone to this problem. Combining the approaches followed by CFRS and CBRS in a single system can alleviate the problems associated with each individual approach.

In summary, the information that is used by a recommender system to produce new recommendations is one of the most contrasting factors of the presented taxonomy.

The systems in each category employ different recommendation methods to produce new recommendations, and make use of different information (ratings, content, or a combination thereof). The recommendation methods usually require numeric representations of items, so proper representation models are needed, as will be detailed in Section 2.5. Moreover, a recommender needs to assess the (dis)similarity between items, so a distance function must be compatible to the numerical representation, as will be described in Section 2.6.

2.2 The property of surprise

In the literature of recommender systems, many works focus on how to discover new items that might be useful to a user. This is a challenging objective and several approaches were proposed to achieve it, as seen in the previous section. In general, these approaches seek for new items that bear some similarity to items which have been given high ratings by some set of users. An even greater challenge is to discover new items that do not resemble items known to a user, yet would still be useful to them. This would be a serendipitous recommendation according to Herlocker, Konstan and Riedl (2002). These authors offer a definition of serendipity that is usually cited by work in this area: "A serendipitous recommendation helps the user find a *surprisingly interesting item* he might not have otherwise discovered." (our emphasis). In a sense, this definition supports a perspective whereby serendipity, seen as a system property, results from the interaction of two other and more fundamental properties: surprise and relevance. Being surprising and relevant (or useful) to a user are the basic requirements of a serendipitous recommendation.

The over-specialisation and the filter bubble conditions, framed as operating states in system parameter space, are associated to a systematic reduction of serendipity in personalised recommendations. The component of surprise in serendipity has a higher association with these conditions than has the relevance because practical personalisation approaches include relevance as their critical property. We focus on surprise because the degree to which a recommender system is exposed to an over-specialised or a filter-bubble condition can be better assessed as a function of how much surprise a recommender system has embedded in its recommendations at two different points in time: $\Delta S = S_{t+1} - S_t$, where S_t is a measure of the system-level surprise at time t. In other words, if relevance

remains reasonably constant between the two measurements while serendipity has decreased in the same time interval, then it must be the case that surprise has also decreased.

However, it has been pointed out by Kaminskas and Bridge (2016) that there is a conceptual overlap between the notions of novelty, unexpectedness, surprise and serendipity in the literature. The metrics for surprise usually assess the degree to which a recommended item differs from the items known to the user, but this notion is also used to define metrics for novelty. For example, Vargas and Castells (2011) proposed several metrics for novelty: some metrics follow a notion of item novelty as being antagonistic to item popularity (popularity-based item novelty), and other metrics follow a notion of novelty as being antagonistic to the user experience (distance-based item novelty); the latter overlaps with the notion of surprise[2]. This overlap is also present in a metric for novelty proposed by Nakatsuji et al. (2010): the metric assesses novelty of a recommended item as its distance to the items known to the user, in which an item is represented as a vector that codes its membership to a set of classes in a taxonomy of items.

To tackle this issue, we clarify the conceptual boundaries by subscribing to the terminology proposed by Kaminskas and Bridge (2016), in which a) novelty is defined only in terms of item popularity so that the conceptual overlap with surprise is removed and novelty can be safely left out of the following analysis; b) unexpectedness usually conveys the same notion as surprise, and thus is be included in the analysis; and c) serendipity has surprise as one of its components, and it is also included in the analysis.

2.3 The metrics for surprise

This section presents a review of six surprise-related metrics in the literature. Taking into account the overlap described in the previous section, this review includes metrics for surprise and metrics for serendipity or unexpectedness whose definitions involve the notion of surprise. For reasons that will be clarified in the next section, we argue that a metric for surprise should address the factors that are related to the validity of a measurement scale used to assess psychological responses of human subjects[3] (JOHN; BENET-MARTINEZ, 2000). We select two of these factors to serve as contrasts between the metrics:

[2] It must be noted that, more recently, these authors have written a book chapter on novelty and diversity in which the notion of novelty as being antagonistic to the user experience has been named as unexpectedness (CASTELLS; HURLEY; VARGAS, 2015).

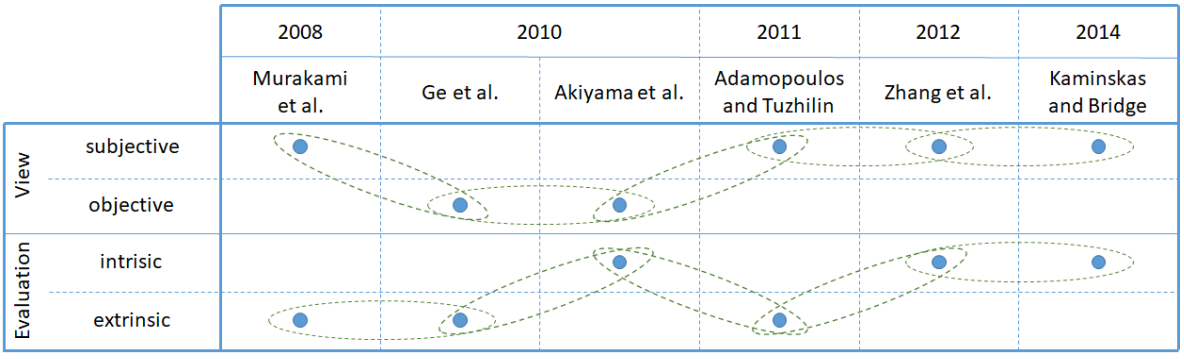
[3] These factors will be later employed to assess the adequacy of a metric from a cognitive perspective.

- *Subjective vs objective view*: some metrics assume that surprise is subjective in nature, since it depends on the user past experience, which usually is represented by the set of items known to a user (MURAKAMI; MORI; ORIHARA, 2008; ADAMOPOULOS; TUZHILIN, 2011; ZHANG et al., 2012; KAMINSKAS; BRIDGE, 2014); other metrics view surprise as a property of the item (GE; DELGADO-BATTENFELD; JANNACH, 2010; AKIYAMA; OBARA; TANIZAKI, 2010) and, thus, is independent of the users.

- *Intrinsic vs extrinsic evaluation*: some metrics only use data that are internal to the system under evaluation (AKIYAMA; OBARA; TANIZAKI, 2010; ZHANG et al., 2012; KAMINSKAS; BRIDGE, 2014); other metrics use data made available by an external system[4] in addition to data that is internal to the system under evaluation (MURAKAMI; MORI; ORIHARA, 2008; GE; DELGADO-BATTENFELD; JANNACH, 2010; ADAMOPOULOS; TUZHILIN, 2011).

A summary of the review is illustrated in Figure 1, which shows how each metric is positioned with regard to these contrasting factors. As can be seen in the diagram, it includes the year of publication of the work in which the metric was proposed, and the ellipses show trends or changes in the factors. This review is not meant to be exhaustive but rather aims to capture how the metrics have evolved over the last decade.

Figure 1 – Evolution of surprise-related metrics over time.



Source: Andre Paulino de Lima, 2019

[4] Such a system is often referred to as baseline system or PPM - Primitive Prediction Model.

2.3.1 A metric for unexpectedness

Murakami, Mori and Orihara (2008) have proposed a metric to evaluate unexpectedness that relies on the ideas that a) recommendations made by a PPM are prone to be obvious, and b) an unexpected recommendation must be non-obvious. As shown in Equation 1, the metric is calculated from a recommendation list L produced for the user u by the system under evaluation. In this definition, the predicate $rscore$ accounts for the predicted relevance of an item L_i to the user u, while $isrel$ accounts for surprise, and reflects the degree to which an item L_i is similar to items rated highly by the user; thus, the metric is aligned to a subjective view of surprise. In fact, Silveira et al. (2017) revisited this metric and implemented $isrel$ as the mean cosine similarity between an item i and items known to the user. It must be noted that the predicate $rscore$, in Equation 2, includes the relevance predicted by both the system under evaluation (Pr) and the external system (Pr^*), and can thus be regarded as an extrinsic evaluation.

$$unexp(L,u) = \frac{1}{|L|}\sum_{i=1}^{|L|} rscore(L_i,u) \times isrel(L_i,u). \tag{1}$$

$$rscore(L_i,u) = max(Pr(L_i,u) - Pr^*(L_i,u), 0). \tag{2}$$

2.3.2 A metric for serendipity

Ge, Delgado-Battenfeld and Jannach (2010) devised a metric for evaluating serendipity that follows the same line of thought pursued by Murakami, Mori and Orihara (2008), although the external system is employed in a different way. As shown in Equation 3, $srdp$ is applied to a recommendation list L^δ, and estimates the *usefulness* of each item L_i^δ, that accounts for relevance. In Equation 4, L^δ is defined as a list that consists of the elements recommended to the user u by the system under evaluation (L) and that do not appear in the list drawn up for user u by an external system (L^*). This means that L^δ comprises non-obvious, unexpected items, and hence only accounts for surprise. It must be noted that $srdp$ operates in an objective way, since estimating surprise (L^δ) does not involve evaluating the degree to which new items are similar to items already known to the user.

$$srdp(L^{\delta}, u) = \frac{1}{|L^{\delta}|} \sum_{i=1}^{|L^{\delta}|} usefulness(L_i^{\delta}, u). \tag{3}$$

$$L^{\delta} = L \backslash L^{*}. \tag{4}$$

2.3.3 A metric for general unexpectedness

Akiyama, Obara and Tanizaki (2010) set out a metric called "general unexpectedness" that explores a combinatorial intuition: an item that shows a rare combination of attributes must be treated as unexpected. It assumes that each item has some content combined with it, and that such a content can be described by a set of attributes. This usually is the case with content-based recommenders (GEMMIS et al., 2015). As shown in Equation 5, the *unexp* metric is estimated for L, the recommendation list produced to user u by the system under evaluation, and this aggregates the *uscore* obtained for each item L_i. The *uscore*, defined in Equation 6, is the reciprocal of the sum of the joint probabilities of pairs of attributes of L_i. In this equation, $A(L_i)$ represents the set of attributes that describe L_i, the expression $a, b \in A(L_i)$ represents the set of all subsets of $A(L_i)$ with two elements, N_a denotes the number of items in the repository that have attribute a and $N_{a,b}$ is the number of items that have both attributes a and b. Thus, an objective view is adopted since surprise can be seen as a property of the content of an item. Differently from surprise metrics that were described earlier, the general unexpectedness does not employ an external system, and thus performs an intrinsic evaluation.

$$unexp(L) = \frac{1}{|L|} \sum_{i=1}^{|L|} uscore(L_i). \tag{5}$$

$$uscore(L_i) = \left[\frac{1}{|A(L_i)|} \sum_{a,b \in A(L_i)} \frac{N_{a,b}}{N_a + N_b - N_{a,b}} \right]^{-1}. \tag{6}$$

2.3.4 A second metric for unexpectedness

Adamopoulos and Tuzhilin (2011) propose a metric for unexpectedness that examines an intuition about user expectation: an item is expected for user u if it is known to them or bears some similarity to items known to them. As shown in Equation 7, the *unexp*

metric is calculated from L, the recommendation list produced for user u by the system under evaluation, and L^s, a list of obvious, expected items that is defined in Equation 8. In that equation, L^* is a recommendation list produced for user u by an external system, E_u represents the set of items that have been rated by user u, and the predicate *neighbours* represents the set of items in the repository (I) that are similar to the items in E_u to some degree specified by threshold parameters in θ. This approach adopts an external system (extrinsic evaluation), and adheres to a subjective view, since it takes account of past experience of the user.

$$unexp(L, L^s) = \frac{1}{|L|}|L \backslash L^s|. \tag{7}$$

$$L^s = L^* \cup E_u \cup neighbours(I, E_u, \theta). \tag{8}$$

2.3.5 The unserendipity metric

Zhang et al. (2012) explore the idea that a serendipitous recommendation must be dissimilar to items known to the user, in a semantic sense. This metric resembles that one proposed by Akiyama, Obara and Tanizaki (2010) in that it requires each item to be combined with some content. However, while the latter employed items represented as sets of attributes, in this metric the content attributes are represented as vectors in $\mathbb{R}^m$. As is shown in Equation 9, the metric is computed from the recommendation list drawn up for user u by the system under evaluation (L), and results in a score that is the average cosine similarity obtained from the items in L and the set of items known to the user (E_u). a) This approach does not employ an external system (intrinsic evaluation), and b) it adheres to a subjective view of surprise. It should be noticed that, unlike the metrics shown earlier, *unsrdp* is scale-inverted, since the lower the score, the more surprising L is.

$$unsrdp(L, u) = \frac{1}{|L||E_u|} \sum_{i \in L} \sum_{j \in E_u} cossim(i, j). \tag{9}$$

2.3.6 A metric for surprise

In a similar way to Zhang et al. (2012), Kaminskas and Bridge (2014) argue that a surprising recommendation must be dissimilar to items known to the user, but does

not require that this dissimilarity should be semantic in nature. They also explore the interplay between the notions of distance and similarity[5]. Equation 10 shows that the metric is calculated from the recommendation list L produced for user u by the system under evaluation, and corresponds to the average surprise obtained for each item in L. The surprise of an item i in L is estimated as either a) the minimum distance between i and each item known to the user (E_u), as described in Equation 11, or b) the maximum degree of similarity between the same items, as shown in Equation 12; in the latter case, the metric is scale-inverted. The predicate *dist* is defined as the Jaccard distance between the set of attributes recovered from contents linked to items i and j (see Section 2.6.3, Equation 22), while the predicate *sim* computes the normalised pointwise mutual information score (NPMI) (BOUMA, 2009) for the joint probability between items[6]. This approach supports a subjective view of surprise, since it takes account of the past experience of the user, and performs an intrinsic evaluation, since does not employ an external system.

$$surprise(L, u) = \frac{1}{|L|} \sum_{i \in L} S_i(i, E_u). \tag{10}$$

$$S_i(i, E_u) = \min_{j \in E_u} dist(i, j). \tag{11}$$

$$S_i(i, E_u) = \max_{j \in E_u} sim(i, j). \tag{12}$$

2.3.7 Summary and analysis

As seen, some metrics subscribe to a subjective view of surprise, while others do not; also, some metrics assume that the information required for assessing surprise must be internal to the system under evaluation and there are metrics that assume otherwise. However, there is a consensus among the metrics described in this review in regard to a fundamental element: all of them involve a notion of distance in their surprise component:

- Murakami, Mori and Orihara (2008): the predicate *isrel* measures how similar an item is to the items known to a user. As mentioned earlier, Silveira et al. (2017) revisited this metric and implemented *isrel* as the mean cosine similarity between an item i and the items known to the user; thus, *isrel* follows a geometric intuition.

[5] Given a distance function, a similarity function can be derived, and vice-versa (DEZA; DEZA, 2009).

[6] This joint probability is estimated as the proportion of users that rated both items i and j.

- Ge, Delgado-Battenfeld and Jannach (2010) : L^{δ} is the difference between sets of items, thus it is a distance based on a combinatorial intuition.
- Akiyama, Obara and Tanizaki (2010): the predicate *uscore* is the reciprocal of a measure of similarity given by joint-probability, thus it may be framed as a distance.
- Adamopoulos and Tuzhilin (2011): the predicate *unexp* is the difference between sets, thus it is a distance based on a combinatorial intuition.
- Zhang et al. (2012): the predicate *unsrdp* is defined as the reciprocal of geometric similarity, and thus may be framed as a distance.
- Kaminskas and Bridge (2014): the predicate S_i directly specifies the distance (of combinatorial intuition) and similarity (of probabilistic intuition) functions.

Despite of this consensus, the metrics produce measurements in scales that are not necessarily comparable with one another, owing to the adoption of different intuitions of distance; in particular, they may disagree on a) the boundaries of the scale, and b) what constitutes a recommendation list that should be mapped to one extreme of the scale:

a) For example, the S_i metric proposed by Kaminskas and Bridge (2014), described in Equation 11, produces measurements within a bounded range (the interval $[0, 1]$), while the metric proposed by Akiyama, Obara and Tanizaki (2010) produces measurements in an unbounded range (the interval $[0, +\infty)$). They agree on the lower bound, but a mapping between their upper bounds is cannot be defined.

b) Even if two metrics agree on their lower bound, that does not imply that they also agree about what constitutes a recommendation list with null surprise. For example, the surprise of a recommendation list L composed of items unknown to a user u is always greater than zero according to the metric in Equation 11, because[7] $L \cap E_u = \varnothing \implies \sum_{i \in L} \min_{j \in E_u} dist(i, j) > 0$. This is not the case with the metric proposed by Adamopoulos and Tuzhilin (2011), described in Equation 7, because $L \cap E_u = \varnothing \wedge L \subset L^* \cup neighbours(I, E_u, \theta) \implies unexp(L, L^s) = 0$.

In summary, despite the consensus about framing surprise as a distance, the argument about the incommensurability between the metrics is supported by divergences on their views (subjective vs objective, intrinsic vs extrinsic) and scales. The minimum and

[7] As will be seen in Section 2.6.1, if $i \neq j$, then $dist(i, j) > 0$, assuming that the data representation model produces distinct item vectors for different items, as will be described in Section 2.5.

the maximum values of a surprise scale may not be attainable by any recommendation list depending on the metric adopted, and the values in between these extremes may reflect significantly different situations, as a result of the adoption of different distance intuitions.

2.4 The experience of surprise

As seen in the previous section, metrics for surprise explore diverse intuitions to model the relationship between the surprise experienced by a user and the recommendation list that was presented to them. In this section, we argue that a metric for surprise would benefit from, and should account for, some ideas that have been explored in the field of cognitive science about how human subjects experience surprise.

In Reisenzein, Horstmann and Schützwohl (2017), it was examined the extent to which the experimental evidence supports current models of surprise in human beings, in particular: the cognitive-evolutionary model (MEYER; REISENZEIN; SCHÜTZWOHL, 1997), and the metacognitive explanation-based model (FOSTER; KEANE, 2015)[8]. According to the cognitive-evolutionary model, the feeling of surprise emerges in response to a failure to anticipate (predict) the occurrence of an "unexpected (schema-discrepant) event" that conveys a change in the environment. In contrast, the metacognitive explanation-based model frames surprise as a response to a failure to track (explain) the cause of a change in the environment. Despite of any divergences, both models presume the existence of a cognitive process that collects and processes information about the environment, which signals surprise when appropriate; it follows from that that all information required to produce the surprise signal must be internal to the cognition of the subject. Moreover, these models agree on the subjective nature of surprise, and, to some extent, both models align to the definition of surprise as a distance, as proposed in Itti and Baldi (2009):

$$S(D, \mathscr{M}) = KL(P(M|D), P(M)) = \int_{\mathscr{M}} P(M|D)\ log\ \frac{P(M|D)}{P(M)}\ dM,$$

where D is an event, $\mathscr{M}$ is the set of models (of the world) held by the subject, KL is the Kullback-Leibler divergence, $P(M)$ is the prior belief assigned to M, and $P(M|D)$ is the belief assigned to M after the event D. As will be seen in Section 2.6, the Kullback-Leibler divergence is a relaxed distance. It must be noted that, in a sense, this definition supports

[8] The review that was cited covers other models, but these two are particularly useful to this analysis.

a perspective whereby surprise corresponds to the amount of information retained by the subject, as the models they hold were adjusted is response to the experienced event.

In view of this, we revisit the contrasting factors from Section 2.3 and apply them to assess the degree of coherence between a metric for surprise and the set of characteristics presumed by the cognitive models. The results of this assessment are summarised in Table 1, and they support the argument that the metric proposed by Kaminskas and Bridge (2014) (Equation 11) is in higher agreement to the cognitive models of surprise because:

1. It adopts a subjective view, unlike (GE; DELGADO-BATTENFELD; JANNACH, 2010; AKIYAMA; OBARA; TANIZAKI, 2010), and it does not depend on information that resides externally to the system being evaluated, unlike (MURAKAMI; MORI; ORIHARA, 2008; ADAMOPOULOS; TUZHILIN, 2011).
2. It accounts for changes in the surprise of an unobserved item, as the growth of the set of items known to the user (E_u), and is more sensitive in this regard than the metric proposed by (ZHANG et al., 2012), because the former assesses the minimum distance between an item and a set, while the latter assesses the average distance.
3. It is proportional to the degree of dissimilarity between a recommended item i and the items known to the user, which embeds a notion of distance, as proposed in Itti and Baldi (2009).

Table 1 – Metrics for surprise in recommender systems and their adherence to characteristics presumed by the two cognitive models. A metric is adherent to the notion of subjectivity if it considers that surprise not only depends on the user experience (column "is subjective"), but also that this experience depends only on information that is endogenous to the user (column "is intrinsic").

	Subjectivity		Sensitivity to user experience
Surprise metric proposed in	Is subjective?	Is intrinsic?	Compared sensitivity
(GE; DELGADO-BATTENFELD; JANNACH, 2010)	no	—	—
(AKIYAMA; OBARA; TANIZAKI, 2010)	no	—	—
(MURAKAMI; MORI; ORIHARA, 2008)	yes	no	—
(ADAMOPOULOS; TUZHILIN, 2011)	yes	no	—
(ZHANG et al., 2012)	yes	yes	lower
(KAMINSKAS; BRIDGE, 2014)	yes	yes	higher

It must be noted that none of the metrics for surprise presented in Section 2.3 accounts for known cognitive biases, and thus they can only provide a rough estimate of the intensity of surprise that is experienced by the user. For example, S_i (Equation 11)

can be seen as an exhaustive search in the user memory (E_u) for similar events. From this perspective, the metric assumes that the user can recall every past event equally well, and thus fails to account for the recall and retrievability biases. The former refers to the relative ease to recall recent and vivid events in relation to events that were observed in a remote past or were unemotional (TVERSKY; KAHNEMAN, 1974; BAZERMAN; MOORE, 2009). The latter refers to how the subjective context can change the relative salience to our perception of the features of an object, and the role that this salience plays in our judgment of similarity (TVERSKY, 1977). There is also a difficulty in conciliating the typical duration of an event. From the exhaustive search analogy, it follows that recommended items must be mapped to events in the terminology of the cognitive models of surprise, but the exposition of a user to an item might take a few minutes (e.g. music, news) up to a couple of hours (e.g. movies, books), whereas the typical duration of an event in the experiments that were analysed in the review of cognitive models of surprise (REISENZEIN; HORSTMANN; SCHÜTZWOHL, 2017) is in the range of seconds. Despite of the limitations just described, and in the absence of metrics of higher fidelity, we argue that this metric is still useful to estimate surprise in recommender systems, and it will be employed to develop the concept of potential surprise in the next chapter.

2.5 The item representation models (or how item vectors are extracted)

In the taxonomy of RS presented in Section 2.1, the information that is used by an RS to produce new recommendations was described as one of the most contrasting factors between categories, and it was also mentioned that a system usually keeps numeric representations of items so that the recommendation method can properly handle the items. In addition, all the surprise metrics presented in Section 2.3 need to assess some form of distance between items. Such numerical representations of items, commonly referred to as item vectors, are obtained by means of a representation model, which determines how different features of the collection of items are combined and encoded in a representation space and how each item is mapped to an element of that space. In the following, we explore the most common item representation models used by the two main categories of recommender systems: the CBRS (Section 2.5.1) and the CFRS (Section 2.5.2).

2.5.1 Item representation in CBRS

In Content-Based Recommender Systems (CBRS), each item is combined with a content that describes some aspects of the item. For example, in a CBRS applied to movie recommendation, each item may be combined with a set of labels, and each label describes a movie genre to which the movie can be associated (e.g. Comedy, Drama, and Romance). In this scenario, the representation space can be defined as the set of all possible sets of genres, and the item is mapped to an element of that space. In this view, two items are similar to the degree that their respective sets of items intersect; this notion of similarity is captured, for example, by the Jaccard similarity function, as will be seen in Section 2.6.

In general, the type of content used to describe an item of a CBRS can include structured and non-structured data:

- *Structured data*: each item is described by a common set of attributes. Each attribute has an intelligible meaning and assumes a single value in its domain. In turn, each domain may be described as a measurement scale, such as those from the statistics tradition (e.g. nominal, ordinal, interval or ratio), though other typologies for data are available (VELLEMAN; WILKINSON, 1993).
- *Non-structured data*: each item is described by a set of features. In this sense, a feature may not have an intelligible meaning, and a set of features that are common to all items may not exist. Data in this category include textual content, images, audio, or video.

It must be noted that structured data usually have a relatively small number of attributes, which are typically curated by domain specialists, whereas unstructured data have a relatively large number of features, and are cheaper to collect and process, especially if made available in a textual format. If the items in a CBRS are combined with textual content, distributional semantic models (DSM) can be used as representation models to produce item vectors, and, for these reasons, we focus on the use of DSM in CBRS.

The Model C

Distributional semantic models are abstractions of the relationship between words and meanings, and are founded on the statistical semantics hypothesis: “statistical patterns

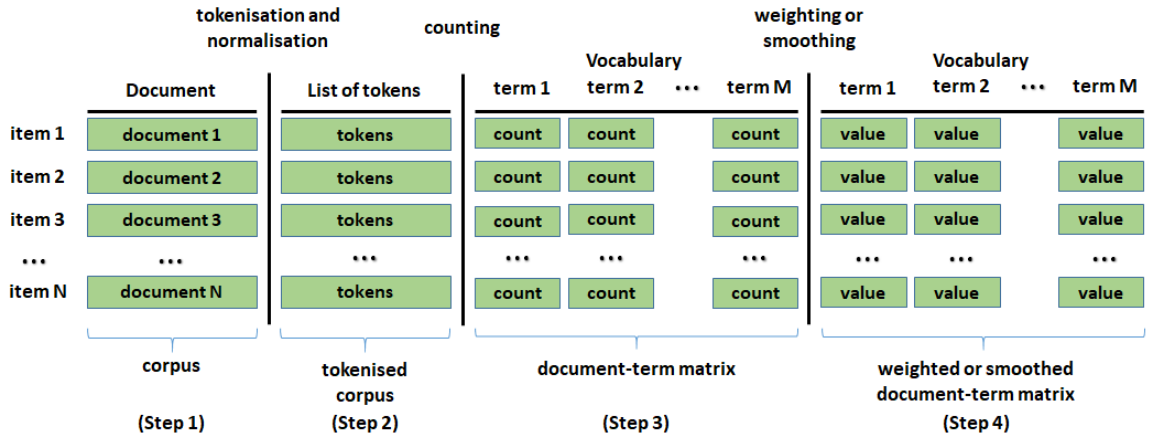
of human word usage can be used to figure out what people mean" (TURNEY; PANTEL, 2010). Over the years, several semantic models that explore this hypothesis have been proposed. One of them, the document-term matrix, which is a type of count-based DSM, is based on an extension of the statistical semantics hypothesis known as the "bag of words hypothesis": two documents tend to convey similar meanings if they have similar counts of the same words (TURNEY; PANTEL, 2010).

To create a document-term matrix that can be used to encode the items in a CBRS with textual content, the four steps illustrated in Figure 2 can be followed (MANNING; RAGHAVAN; SCHÜTZE, 2008):

- Step 1: create a corpus composed of the textual content that is combined with each item. Following the usual approach in information retrieval, consider each textual description as a document: if the recommender system has N items in its repository, then the corpus must contain N documents.
- Step 2: apply tokenisation and appropriate text normalisation transformations (e.g. stop words removal, stemming, lemmatisation, case folding) to each document. After these transformations, each document in the corpus is represented by a list of tokens in a new, tokenised corpus that also contains N documents.

- Step 3: create the corpus vocabulary as the set of tokens that occur in the tokenised corpus. Each token in the vocabulary is referred to as a term. After creating the vocabulary, count the number of occurrences of each term in a document, and create a

Figure 2 – Creating item vectors using a count-based DSM.



Source: Andre Paulino de Lima, 2019

vector with the counts obtained for each term. The matrix composed of the resulting count vectors is called a document-term matrix, because each row is related to the original document associated to an item, and each column represents a term that appears in one or more documents in the corpus.

- Step 4: apply weighting (e.g. tf-idf weighting) or smoothing (e.g. Bayesian multiplicative treatment) as appropriate. The weighting transformation aims to calibrate the relevance of the terms that occur frequently in the corpus in relation to terms that occur less frequently. The smoothing transformation aims to eliminate the sparsity of the resulting document-term matrix by replacing zero-valued elements with some small, non-zero value, and adjusting the remaining elements in a way that the ratio between any two non-zero elements is preserved as much as possible (EGOZCUE et al., 2011). There are other alternative approaches for smoothing, such as dimensionality reduction or feature selection, but these approaches involve making decisions about how much information can be lost in the transformation (TURNEY; PANTEL, 2010).

In this model, which is referred to hereafter as **Model C**, an item is represented by its corresponding row vector in the document-term matrix. Thus, if the corpus vocabulary has m terms, then items will be represented as m-dimensional vectors. If the document-term matrix is not weighted or smoothed, then all elements in an item vector will be non-negative integers, since each element represent a term count. If the document-term matrix is weighted or smoothed using methods mentioned in Step 4, then all elements in an item vector will be non-negative real values. If the document-term matrix is weighted or smoothed using alternative approaches, then elements may assume negative real values. As will be seen in Section 2.6, this characteristic is important because a similarity or distance function imposes requirements on the structure of the item vectors to which it is applied.

It must be noted that an item vector produced by the Model C describes the composition of a unit of text in terms of how many occurrences of each word (or component) is there. This aspect is important because there are known issues that arise when analysing data that describe compositions, which are also known as compositional data in statistics. According to the literature, compositional data consist of vectors whose components can be framed as the proportion or percentages of some whole (AITCHISON, 2003). For example, suppose that a matrix $\mathbf{V}$ describes the composition of a set of objects as row vectors $\mathbf{v}_i$,

and its columns represent different components of such objects. In addition, suppose that the correlation between any two components is zero, so knowing the quantity v_{ij} does not inform about the quantity v_{ik} in any vector. In this scenario, if normalisation to unit sum is applied to the vectors in $\mathbf{V}$ (i.e. dividing each vector element v_{ij} by the sum of all the elements in that vector, $\sum_k v_{ik}$), then spurious correlations will arise between the components of $\mathbf{V}$ (AITCHISON, 2003). Some distance functions described in Section 2.6 require that the vectors should be normalised to unit sum, and transformations will then be described to avoid issues when handling compositional data.

The Model D

As mentioned earlier, several semantic models that explore the statistical semantics hypothesis have been proposed over the years. As reported in Baroni, Dinu and Kruszewski (2014), one of such models, known as predictive DSM , has been attracting the attention of many research communities in recent years. It is based on another extension of the statistical semantics hypothesis, referred to as the "distributional hypothesis": "words that occur in similar contexts tend to have similar meanings" (TURNEY; PANTEL, 2010).

A predictive DSM can be obtained from a language model that has been optimised to perform a specific predictive task. However, before describing how to encode items in a CBRS by using a predictive DSM, the concept of "language model" and the kind of "predictive task" it can perform must be clarified.

Language models

A language model is a probability distribution over a set of terms in a vocabulary that can be used to solve the task of predicting the next word in a sentence extracted from a corpus (MANNING; RAGHAVAN; SCHÜTZE, 2008; BENGIO et al., 2003). In its simplest form, a language model M_1 over the vocabulary ν is a distribution P such that:

$$\sum_{w \in \nu} P(w) = 1, \text{where } w \text{ is a term of the vocabulary } \nu.$$

In this model, the probability assigned to a term w is independent of the words that appear before or after it in any sentence. However, to be able to explore the distributional hypothesis, a language model must consider the notion of "context of a word". This context

is usually taken as the sequence of words that precedes a particular term in sentences extracted from a corpus[9]. For example, a language model M_2 over the vocabulary ν may be described as a conditional distribution P such that:

$$\sum_{w_t, w_{t-1} \in \nu} P(w_t|w_{t-1}) = 1, \text{where } t \text{ indicates the position of a word } w \text{ within the context.}$$

In this model, the probability assigned to a term w_t is conditioned on the words that precede it in the corpus, which are identified by the expression w_{t-1}. Here, the context (w_{t-1}, w_t) is any sequence of two consecutive words in a corpus in which the last word is the term w_t. The model M_2 is called as a bigram language model because its context consists of two words, whereas M_1 is referred to as unigram language model.

It is possible to generalise the definition of a language model for contexts of arbitrary length: a language model M_n over the vocabulary ν, which has a context of length n, is a conditional distribution P such that:

$$\sum_{w_{t-n+1}, \ldots, w_t \in \nu} P(w_t|w_{t-n+1}^{t-1}) = 1,$$

where w_{t-n+1}^{t-1} is a sequence $n-1$ words that precede any occurrence of w_t in the corpus. Here, the context is any sequence of n words ending in w_t, namely $(w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}, w_t)$.

Language model and vector representation of words and documents

Given a corpus, the process of estimating a conditional distribution P of a language model M basically involves two decisions: which context size and which estimation process to adopt. Although the context size usually is selected empirically, there are many alternatives to estimate P. Mikolov et al. (2013a), improving ideas previously explored by Bengio et al. (2003), framed the problem of estimating P as a problem of finding a function $\hat{P}$ that approximates P by using a neural network model. As a by-product of the process of finding an approximate conditional distribution $\hat{P}(w_t|w_{t-n+1}^{t-1})$, the neural network model produces a matrix $\mathbf{C}$ with the following characteristics:

- $\mathbf{C}$ is a $|\nu| \times d$ matrix, where $|\nu|$ is the number of terms in the corpus vocabulary and d is a free parameter of the neural network model.

[9] Although it is possible to define a language model that defines its context as a sequence of words that contain a particular term, comprising both words preceding and succeeding the term.

- The i^{th} row of $\mathbf{C}$, namely $\mathbf{c}_i$, is a d-dimensional vector that can be used as a vector representation for the term w_i.
- A vector $\mathbf{c}_i$ captures semantic and syntactic features of the term w_i, and can be used in tasks that involve assessing semantic or syntactic similarity between terms.

An extension to this approach has been proposed by Mikolov et al. (2013b), in which a neural network model can be used to produce vector representations of units of text larger than a word (e.g. sentence, paragraph, document). Estimating P is also framed as finding a function $\hat{P}$ that approximates P and, as a by-product of the process, the neural network model produces an additional matrix $\mathbf{D}$ with the following characteristics:

- $\mathbf{D}$ is a $|corpus| \times d$ matrix, where $|corpus|$ is the number of documents in the corpus and d is a free parameter of the neural network model.
- The i^{th} row of $\mathbf{D}$, namely $\mathbf{d}_i$, is a d-dimensional vector that can be used as a vector representation for the document i.
- The vector $\mathbf{d}_i$ captures topical and semantic features of the document i and have being shown to perform well in tasks that involve assessing semantic similarity between documents of varying lengths (DAI; OLAH; LE, 2015; JAWAHAR, 2017).

Obtaining item vectors from a predictive DSM

To obtain item vectors using a predictive DSM for items combined with textual content, the three steps illustrated in Figure 3 can be followed:

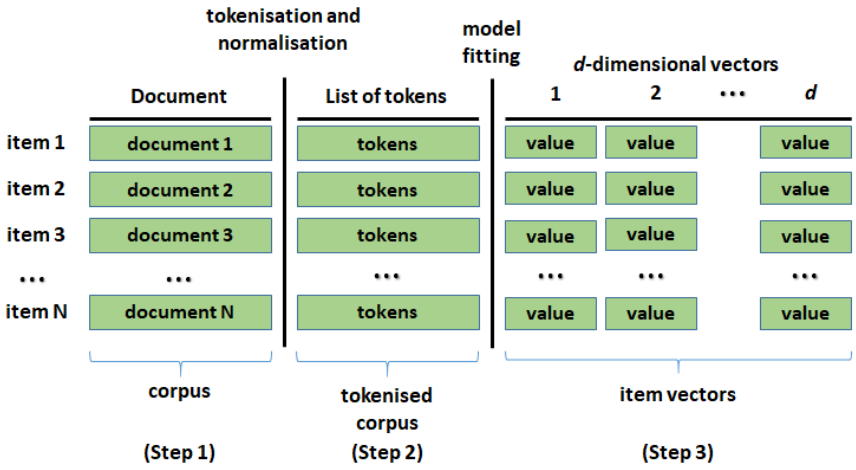
- Step 1: create a corpus composed of the textual content that is combined with each item. Treat each textual description as a document: if the RS has N items in its repository, then the corpus will contain N documents.

- Step 2: apply tokenisation and appropriate text normalisation transformations. Unlike the count-based DSM, stop words removal is not applied because this normalisation operation changes the context where a word appears and, as a result, affect the ability of the model to capture semantic and syntactic features from the text. In addition, end of sentence punctuation must be preserved. After the transformations,

each item in the corpus is represented by a list of tokens in a new, tokenised corpus.

- Step 3: instantiate a predictive DSM and fit it to the corpus. There are several open source implementations of predictive DSM that can be considered for this task (HILL; CHO; KORHONEN, 2016; JAWAHAR, 2017). Finally, as the result of the optimisation, a matrix is created containing a vector representation for each document in the corpus.

Figure 3 – Creating item vectors using a predictive DSM.



Source: Andre Paulino de Lima, 2019

Unlike the count-based models, the sparsity of item vectors produced by a predictive DSM is indirectly controlled by a free parameter that determines the number of dimensions d, which is defined during the optimising process[10]. Thus, smoothing is usually not required. Also unlike count-based models, vector elements may assume positive or negative real values. Since compositional data consist of vectors whose components are framed as the proportion or percentages of some whole (AITCHISON, 2003), item vectors with negative elements cannot be considered compositional data. Besides, the presence of negative elements forbids the use of a similarity or distance function that requires item vectors with non-negative elements, as will be seen in Section 2.6.

Another important aspect of this model, which is referred to hereafter as **Model D**, is that it can be cast as a lower dimensional model obtained from the document-term matrix (LEVY; GOLDBERG, 2014). The dimensionality reduction results from applying stochastic gradient descent (SGD) to optimise the objective function of a neural model.

[10] In machine learning, the optimising process is commonly referred to as model fitting or model training.

2.5.2 Item representation in CFRS

As mentioned in Section 2.1, Collaborative Filtering Recommender Systems (CFRS) exclusively employ ratings to produce new recommendations. Systems in this category usually organise all the ratings in a user-item matrix $\mathbf{R}$[11], in a such that each element r_{ui} stores the rating given by user u to item i. Item vectors can be extracted directly from the user-item matrix (Model U), or from a factorised model based on it (Model V).

The Model U

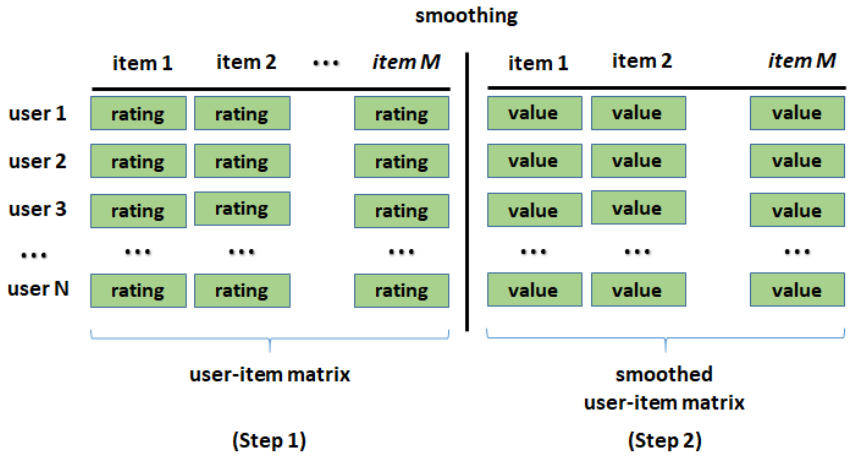
Unlike the previously described models, items vectors correspond to the columns of the user-item matrix (NING; DESROSIERS; KARYPIS, 2015), which can be obtained by executing the two following steps, which are also illustrated in Figure 4:

- Step 1: create a $|U| \times |I|$ matrix $\mathbf{R}$, where U is the set of users and I is the set of items. Fill in each element of $\mathbf{R}$, namely r_{ui}, with the rating the user u has given to item i, or with zero, if the item was not rated by the user. A vector representation for the item i is the i^{th} column vector of $\mathbf{R}$, meaning that the item is represented by a tuple of the ratings attributed to it by all users in the system. It is common to apply some normalisation after the matrix is created, but we postpone this operation to the recommendation method, with the intent of obtaining more flexibility in the experiments that will be reported in the next chapter.

- Step 2: apply smoothing as appropriate. The smoothing transformation aims to eliminate the sparsity of the resulting matrix by replacing zero-valued elements with some small, non-zero value and adjusting the remaining elements in a way that the ratio between any two non-zero elements is preserved as much as possible (EGOZCUE et al., 2011). Similar to the smoothing transformations applicable to count-based DSMs, there are other alternative approaches for smoothing, such as a) matrix factorisation, which reduces dimensionality (KOREN; BELL, 2015), or b) replacing a zero-valued element r_{ui} with an estimated rating $\hat{r}_{ui}$ obtained by using a neighbourhood-based recommendation method (NING; DESROSIERS; KARYPIS, 2015).

[11] The user-item matrix is commonly referred to as rating matrix in the literature, but we adopt this nomenclature to point out its dimensions, as we did with the document-term matrix.

Figure 4 – Creating item vectors using a user-item matrix.



Source: Andre Paulino de Lima, 2019

In this model, which is referred to hereafter as **Model U**, an item vector is $|U|$-dimensional. Assuming the rating scale can be mapped to a non-negative integer interval, then all elements in an item vector will be non-negative integers since each element represents a user rating. If the user-item matrix is smoothed using the method mentioned in Step 2, then all the elements will be non-negative real values. Alternative smoothing methods may introduce negative real values in the resulting matrix. As will be seen in Section 2.6, the domain where elements take their values is important because a similarity or distance function applied to item vectors may impose some requirement on their structure. Moreover, since compositional data consist of vectors whose components can be framed as the proportion of some whole (AITCHISON, 2003), item vectors extracted from user-item matrices can be considered compositional data because each vector element r_{ui} can be cast as a proportion of the total "rating points" given to item i, namely $\sum_{u \in U} r_{ui}$.

The Model V

As mentioned earlier, item vectors can also be extracted from a factorised model obtained by applying singular value decomposition (SVD) to the user-item matrix (CREMONESI; KOREN; TURRIN, 2010; KOREN; BELL, 2015). This process consists of the three following steps:

- Step 1: create a user-item matrix **R** as described in the Step 1 from Model U.

- Step 2: apply SVD to the $\mathbf{R}$ matrix to recover the f dimensions with higher variance. This step produces three matrices that can reconstruct the matrix $\mathbf{R}$ to some extent:

$$\mathbf{R} \approx \mathbf{U} \cdot \Sigma \cdot \mathbf{Q}^T. \tag{13}$$

- Step 3: extract the item vectors from the columns of the matrix $\mathbf{Q}^T$. The factorisation can be seen as the mapping of a raw user-item (column) vector $\mathbf{r}_i$ to a (column) vector $\mathbf{q}_i^T$ that resides in a latent factor space. This point will be further elaborated in Section 2.7, in which the PureSVD recommendation algorithm is detailed.

Unlike the Model U, the sparsity of item vectors from **Model V** is indirectly controlled by a free parameter that determines the number of dimensions f of the latent space, which is defined during the solution of the SVD task; thus, smoothing is usually not required. Also unlike the Model U, vector elements may assume positive or negative real values, which precludes item vectors from Model V to be processed as compositional data, and there are additional restrictions to which distance functions can be applied to them.

2.5.3 Summary and analysis

This section presented four item representation models: Models C and D can produce item vectors from the textual content that is combined with each item and thus can be applied to a CBRS; Models U and V can produce item vectors from ratings in the system and thus is adequate for CFRS. Model D can be considered as a factorised model obtained from Model C (document-term matrix), just as the Model V can be considered as a factorised model obtained from Model U (user-item matrix). For this reason, Models C and U are later referred to as sparse models, and Models D and V are referred to as dense models. We pointed out that Models C and U are represented by matrices with non-negative elements that can be framed as proportions of a whole; this fact qualifies both models as holding compositional data. The same cannot be stated about Models D and V, since they are represented by matrices that can contain negative elements. In addition, it was made clear that the presence of negative elements in an item vector may preclude its use with some distance functions, as will be seen in the next section.

2.6 The item comparison functions (or how item vectors are compared)

In the taxonomy presented in Section 2.1, it was shown that most categories of RS keep item vectors because their recommendation method can only handle numerical representations, and more often than not, they are required to assess the (dis)similarity between two arbitrary item vectors. Of equal importance to this project is the consensus among different authors about modelling surprise as a distance between item representations, though there is no agreement on which specific distance function must be employed in a surprise metric, as was seen in Section 2.3. Moreover, the two cognitive models of surprise presented in Section 2.4 align, to some extent, to the definition of surprise proposed by Itti and Baldi (2009), which uses the Kullback-Leibler divergence to model the relationship between surprise intensity, the user experience, and an event, as seen in Section 2.4.

In the absence of, or maybe in the impossibility of, a deterministic distance function that can provide the metric with an adequate fidelity, we selected a set of distance functions, from varying intuitions, that can be used to compare item vectors from the representation models that were described in Section 2.5. We explore four distance intuitions: *geometric*, which includes the Euclidean and the cosine distances; *combinatorial*, which includes the Jaccard distance; *informational*, which comprises the Kullback-Leibler and the Jensen-Shannon divergences; and *compositional*, which includes the Aitchison distance.

The remainder of this section is organised as follows: in Section 2.6.1, definitions for metric, distance and similarity are provided, and the relationship between these concepts is formalised. Distance functions of diverse intuitions are presented in Sections 2.6.2 to 2.6.5, along with an analysis of their requirements on the structure of the item vectors being compared. Finally, a discussion on the applicability of each distance function in item vectors extracted from different models is summarised in Section 2.6.6.

2.6.1 Metric, distance and similarity

The term “distance” may assume distinct meanings depending on the context it appears. For example, we may use it to refer to the distance between two places in a city, or the distance between two cities. In the first case, it denotes the length of a line segment that connects two points over a plane, or the sum of the length of a collection of line segments. In the second case, it denotes the length of a curve over the surface of an ellipsoid

object, if the two cities in question are Tokyo and São Paulo. The example illustrates that the term may denote distinct forms of separation within a domain (distance between places in cities). However, these forms always have common grounds. For example, if two objects are in the same "place", then they are not separated, and the distance between them is null. In a more abstract analogy, if two descriptions presumably recovered from distinct objects are indiscernible from one another, then all of their (observed) features are equal, no difference between them can be pointed out and, thus, they are perfectly similar to one another. The concept of metric, a fundamental abstraction from the discipline of Metric Spaces (O'SEARCOID, 2007), captures these notions in a general mathematical construct[12]:

Definition 2.6.1 *A metric for a set X is a real function $d\colon X \times X \to \mathbb{R}_+$ that satisfies the following four axioms:*

A1) $\forall\, a, b \in X,\ d(a,b) \geq 0$ *(non-negativity).*
A2) $\forall\, a, b \in X,\ d(a,b) = d(b,a)$ *(symmetry).*
A3) $\forall\, a, b \in X,\ d(a,b) = 0 \iff a = b$ *(identity of indiscernibles).*
A4) $\forall\, a, b, c \in X,\ d(a,b) \leq d(a,c) + d(c,b)$ *(triangle inequality).* ■

In this definitional system, metric is the function d and distance is the real value $d(a,b)$ produced by the metric d when applied to any two arbitrary elements of X. However, it is possible to find other definitional systems (DEZA; DEZA, 2009), in which the concept of distance is confounded with the concept of metric, and sometimes it is not required to satisfy all four axioms. For example, a distance can be defined as a real function $d'\colon X \times X \to \mathbb{R}$ that only satisfies the axioms A1, A2 and A3.

Analogously, the concept of similarity is also defined as a real function over an arbitrary set X (DEZA; DEZA, 2009):

Definition 2.6.2 *A similarity for a set X is a real function $s\colon X \times X \to \mathbb{R}_+$ that satisfies the following four axioms:*

B1) $\forall a, b \in X,\ s(a,b) \geq 0$ *(non-negativity).*
B2) $\forall a, b \in X,\ s(a,b) = s(b,a)$ *(symmetry).*

[12] It must be noted that the functions described in Section 2.3 are called metrics in the literature of RS, but this usage of the term may not agree to the concept of metric commonly used in Mathematics.

B3) $\forall a, b \in X,\ s(a,b) = s(a,a) \iff a = b$ *(identity of indiscernibles).*

B4) $\forall a, b \in X,\ s(a,b) \leq s(a,a)$. ■

It must be noted that, unlike the definition of distance, a similarity function has a well-defined upper bound, which is given by $sup\{s(a,a) \colon a \in X\}$.

In despite of the fact that the terms "distance" and "similarity" express opposite notions in their colloquial usage (i.e., how far vs how close, difference vs sameness, discrepancy vs conformity, dissensus vs consensus, awayness vs nearness), the formal definitions presented here make them intrinsically linked to one another. For example, suppose that a similarity function s' assigns the real value $C > 0$ to any identity, so that $\forall a, b \in X, 0 \leq s'(a,b) \leq s'(a,a) = s'(b,b) = C$. In this case, it is possible to derive a distance function $d' \colon X \times X \to \mathbb{R}_+$ by applying different monotonic mappings:

$$\begin{aligned} &d'(a,b) \mapsto C - s'(a,b), \\ &d'(a,b) \mapsto \frac{C - s'(a,b)}{C}, \text{ or} \\ &d'(a,b) \mapsto \sqrt{C - s'(a,b)}. \end{aligned} \tag{14}$$

Analogously, it is also possible to define a similarity function s' from a distance function d' by exploring different monotonic mappings:

- if d' is a bounded function such that $\forall a, b \in X, d'(a,b) <= C$, then:

$$\begin{aligned} &s'(a,b) \mapsto C - d'(a,b), \text{ or} \\ &s'(a,b) \mapsto \frac{C - d'(a,b)}{C}. \end{aligned} \tag{15}$$

- if d' is an unbounded function, then s' such that $\forall a \in X, s'(a,a) = C$ can be derived by the following mapping:

$$s'(a,b) \mapsto C\left(1 - \frac{d'(a,b)}{1 + d'(a,b)}\right) = \frac{C}{1 + d'(a,b)}. \tag{16}$$

As mentioned earlier, the concept of distance conveys notions of difference or separation, which may have different interpretations depending on how different object features are encoded in the representation space. In Section 2.5, several models that can be used to create item vectors were presented, and the following sections describe how different intuitions can be used to assess the similarity between vectors. To avoid confusion over the terms metric and distance, we hereafter use the term "**distance function**" to refer to a function that satisfies the axioms A1 (non-negativity) and A3 (identity of indiscernibles).

2.6.2 Distance functions based on geometric intuition

Functions in this category use geometric features of the vectors to compute distance between them. For example, the Euclidean distance (also known as L_2 metric), which produces a value that corresponds to the length of the line segment linking two points in a (hyper)plane, is defined as:

$$\begin{aligned} &euclidist\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+ \\ &\qquad (\mathbf{v}, \mathbf{w}) \mapsto ||\mathbf{v} - \mathbf{w}||_2 = \left(\sum_{i=1}^{n} (v_i - w_i)^2 \right)^{\frac{1}{2}}. \end{aligned} \tag{17}$$

Another example of function of geometric intuition is the cosine similarity, which is commonly used in the literature on information retrieval to assess semantic similarity between units of text of different granularities, such as words (JURAFSKY; MARTIN, 2018), sentences (CER et al., 2017), or entire documents (MANNING; RAGHAVAN; SCHÜTZE, 2008). The cosine similarity produces a value that corresponds to the angle between two vectors, which is defined as the inner product between vectors normalised to unit length:

$$\begin{aligned} &s\colon \mathbb{R}^n \times \mathbb{R}^n \to [-1, 1] \subset \mathbb{R} \\ &\qquad (\mathbf{v}, \mathbf{w}) \mapsto \langle \frac{\mathbf{v}}{||\mathbf{v}||_2}, \frac{\mathbf{w}}{||\mathbf{w}||_2} \rangle. \end{aligned} \tag{18}$$

However, this function is not a similarity according to the Definition 2.6.2, because its range is the real interval $[-1, 1]$ and, thus, violates the axiom B1 (non-negativity). Nonetheless, a proper similarity function can be derived from the Equation 18 by applying translation and (optionally) rescale transformations:

$$\begin{aligned} &cossim\colon \mathbb{R}^n \times \mathbb{R}^n \to [0, 1] \subset \mathbb{R}_+ \\ &\qquad (\mathbf{v}, \mathbf{w}) \mapsto \frac{1}{2} \left(1 + \langle \frac{\mathbf{v}}{||\mathbf{v}||_2}, \frac{\mathbf{w}}{||\mathbf{w}||_2} \rangle \right). \end{aligned} \tag{19}$$

Then, a cosine distance can be derived by using the mapping in Equation 14:

$$\begin{aligned} &cosdist\colon \mathbb{R}^n \times \mathbb{R}^n \to [0, 1] \subset \mathbb{R}_+ \\ &\qquad (\mathbf{v}, \mathbf{w}) \mapsto 1 - cossim(\mathbf{v}, \mathbf{w}) = \frac{1}{2} \left(1 - \langle \frac{\mathbf{v}}{||\mathbf{v}||_2}, \frac{\mathbf{w}}{||\mathbf{w}||_2} \rangle \right). \end{aligned} \tag{20}$$

2.6.3 Distances based on combinatorial intuition

In representation spaces where an element is represented by a set of attributes, it is possible to define a similarity that explore how these attributes match between objects. Lee (1999) considers that such a function follows a combinatorial intuition: the greater the number of attributes in common, the greater the similarity between them. This is the case of the Jaccard similarity, whose original definition corresponds to the proportion of common items between two sets (JACCARD, 1912):

$$\begin{aligned} sim\colon U \times U &\to [0,1] \subset \mathbb{R}_+ \\ (X,Y) &\mapsto \frac{|X \cap Y|}{|X \cup Y|}, \end{aligned} \tag{21}$$

where U is a power set of some arbitrary, non-empty set.

In Equation 12, a surprise metric proposed by Kaminskas and Bridge (2014) uses a distance derived from the Jaccard similarity by using the mapping in Equation 14:

$$\begin{aligned} dist\colon U \times U &\to [0,1] \subset \mathbb{R}_+ \\ (X,Y) &\mapsto 1 - \frac{|X \cap Y|}{|X \cup Y|}. \end{aligned} \tag{22}$$

In Jurafsky and Martin (2018), the authors describe an adaptation of the Jaccard similarity that can be applied to real vectors:

$$\begin{aligned} jaccardsim\colon \mathbb{R}_+^n \times \mathbb{R}_+^n &\to [0,1] \subset \mathbb{R}_+ \\ (\mathbf{v},\mathbf{w}) &\mapsto \frac{\sum_{i=1}^{n} min(v_i, w_i)}{\sum_{i=1}^{n} max(v_i, w_i)}. \end{aligned} \tag{23}$$

The idea behind this adaptation is that each element of a vector $\mathbf{v}$ encodes the intensity of a continuous attribute that may vary from complete absence to full presence. Formally, given two vectors $\mathbf{v}$ and $\mathbf{w}$, it is the case that $v_i, w_i \in [0, max\{v_i, w_i\}], \forall i = 1 \ldots n$. If $v_i = 0$, then the attribute associated to the dimension i is absent in the object represented by $\mathbf{v}$; if $v_i = max\{v_i, w_i\}$, then the attribute is fully present in that object. The min and max operations in Equation 23 are used as proxies for intersection and union, which are the operations used in Equation 21, so that $[0, v_i] \cap [0, w_i] = [0, min\{v_i, w_i\}]$ and $[0, v_i] \cup [0, w_i] = [0, max\{v_i, w_i\}]$. In view of this, vectors are constrained to $\mathbb{R}_+^n$ because

negative elements would imply a valid interpretation for an attribute with negative intensity, which is not supported by the "absence vs presence" dichotomy explored by this distance.

Analogously, a Jaccard distance can also be derived from *jaccardsim* by using the mapping in Equation 14:

$$\begin{aligned} jaccardist\colon \mathbb{R}_+^n \times \mathbb{R}_+^n &\to [0,1] \subset \mathbb{R}_+ \\ (\mathbf{v}, \mathbf{w}) &\mapsto 1 - \frac{\sum_{i=1}^{n} min(v_i, w_i)}{\sum_{i=1}^{n} max(v_i, w_i)}. \end{aligned} \tag{24}$$

2.6.4 Distances based on informational intuition

The distance functions presented in this section capture a notion of difference from an informational perspective: two probability distributions are similar to the extent that they concentrate probability mass on the same points. Thus, a first step in assessing the informational difference between two item vectors is to define a transformation that converts them into probabilistic distributions. For item vectors extracted from models that produce compositional data, such as Model C (document-term matrix, Section 2.5.1) and Model U (user-item matrix, Section 2.5.2), the transformation must convert them into probabilistic distributions, and assure that additional requirements imposed by the distance function are satisfied. As for item vectors from Models D and V, which contain non-compositional data, they cannot be safely compared by informational distances because the requirements imposed by these distances are not met.

Converting compositional data into probabilistic distributions

A transformation that converts compositional item vectors into probabilistic distributions generally involves two steps: a) estimating a raw probability distribution for the item vector, and b) replacing zeros in the distribution with small non-negative values. A

raw distribution is traditionally obtained by applying Maximum Likelihood Estimation (MLE) to the item vector (BIGI, 2003), which can be described as:

$$\begin{aligned} MLE\colon \mathbb{R}_{+}^{n} &\to \mathbb{R}_{+1}^{n} \\ \mathbf{v} &\mapsto \frac{\mathbf{v}}{||\mathbf{v}||_1} = \frac{1}{\sum_{i=1}^{n} v_i}\,\mathbf{v}, \end{aligned} \tag{25}$$

where $\mathbb{R}_{+1}^{n} = \{\mathbf{v} \in \mathbb{R}^n; v_i \geq 0, \forall i = 1 \ldots n; \sum_{i=1}^{n} v_i = 1\}$.

It must be noted that the transformation is only defined for vectors whose elements are non-negative real values. This transformation corresponds to a normalisation to unit sum (L_1 norm), and is also called as **unit closure** operation in compositional data (EGOZCUE et al., 2011), because the sum of the elements in the resulting vector is one:

$$\begin{aligned} \mathcal{C}\colon \mathbb{R}_{+}^{n} &\to \mathbb{R}_{+1}^{n} \\ \mathbf{v} &\mapsto \frac{\mathbf{v}}{||\mathbf{v}||_1}. \end{aligned} \tag{26}$$

The final step, which consists of replacing the zero-valued elements of the closed vector (normalised to unit sum) with a small value without violating the restriction that $\sum_i v_i = 1$, may be achieved by applying Bayesian Multiplicative Treatment (BMT) with Perks prior (MARTÍN-FERNÁNDEZ; PALAREA-ALBALADEJO; OLEA, 2011)[13,14]:

$$\begin{aligned} BMT\colon \mathbb{R}_{+1}^{n} &\to \mathbb{S}^{n} \\ \mathbf{v} \mapsto \mathbf{v}' = (v_1', v_2', \ldots, v_n'), v_i' &= \begin{cases} v_i\big(1 - \sum_{k|v_k=0} \frac{\alpha}{T+s}\big), & \text{if } v_i > 0, \\ \frac{\alpha}{T+s}, & \text{if } v_i = 0, \end{cases} \end{aligned} \tag{27}$$

where $\alpha = \frac{1}{n}$, $s = 1$, $T = \sum_i v_i$, and the expression $k|v_k = 0$ means the set of indices of $\mathbf{v}$ such that $v_k = 0$. The codomain of BMT is the unit n-simplex, $\mathbb{S}^n$, defined as:

$$\mathbb{S}^{n} = \{\mathbf{v}' \in \mathbb{R}^n; v_i' > 0, \forall i = 1 \ldots n; \sum_{i=1}^{n} v_i' = 1\}. \tag{28}$$

[13] As described in Martín-Fernández, Palarea-Albaladejo and Olea (2011), it is possible to adopt different parameters (α and s) for each component and instance if domain information is available.

[14] The Perks prior produces smaller values for w_i than other priors (e.g. Bayes-Laplace, Jeffreys or Haldane) do when $v_i = 0$. Thus, the adoption of the Perks prior promotes higher discriminability in similarity assessments because it increases the average difference between element values.

The Kullback-Leibler divergence

Given that the item vectors have been transformed into probabilistic distributions, the difference between them can be calculated by using the Kullback-Leibler divergence:

$$kldiv\colon \mathbb{S}^n \times \mathbb{S}^n \to \mathbb{R}_+$$
$$(\mathbf{v}, \mathbf{w}) \mapsto \sum_{i=1}^{m} v_i \, ln\big(\frac{v_i}{w_i}\big). \tag{29}$$

This function requires the vectors to be normalised to unit sum and free of zero-valued elements; these requirements were satisfied by the closure and smoothing operations previously described. It must be noted that this distance function does not satisfy the symmetry and triangular inequality axioms (A2 and A4, respectively).

The Jensen-Shannon divergence

The Jensen-Shannon divergence is defined as a composition of the Kullback-Leibler divergence which is endowed with symmetry:

$$jsdiv\colon \mathbb{S}^n \times \mathbb{S}^n \to \mathbb{R}_+$$
$$(\mathbf{v}, \mathbf{w}) \mapsto kldiv(\mathbf{v}, \frac{\mathbf{v}+\mathbf{w}}{2}) + kldiv(\mathbf{w}, \frac{\mathbf{v}+\mathbf{w}}{2}). \tag{30}$$

2.6.5 Distances based on compositional intuition

The distance function presented in this section captures a notion of separation that has been explored in statistical analysis of compositional data. The idea is that, in some settings, the ratio between the components of a vector may be more effective to capture the notion of separation than other intuitions.

The approach followed in compositional data analysis (CoDA) (EGOZCUE et al., 2011) to compute the distance between two item vectors requires that: a) the vectors should be transformed into its simplex representations by applying the closure and BMT transformations described in Section 2.6.4, and b) the resulting vectors should be transformed into a representation that is compatible to a metric in the simplex space.

The Aitchison distance is a metric in the simplex space and is defined as the Euclidean distance (Equation 17) applied to transformed simplex vectors (EGOZCUE et al., 2011):

$$\begin{aligned} aitdist &: \mathbb{S}^n \times \mathbb{S}^n \to \mathbb{R}_+ \\ &(\mathbf{v}, \mathbf{w}) \mapsto euclidist(clr(\mathbf{v}), clr(\mathbf{w})), \end{aligned} \tag{31}$$

where the *clr* is the centred log-ratio representation (clr) of a simplex vector, defined as:

$$\begin{aligned} clr &: \mathbb{S}^n \to \mathbb{R}^n \\ &\mathbf{v} \mapsto \mathbf{v}' = (v'_1, v'_2, \ldots, v'_n), v'_i = log(v_i) - log\left[\left(\prod_{i=1}^{n} v_i \right)^{\frac{1}{n}} \right]. \end{aligned} \tag{32}$$

2.6.6 Summary and analysis

To apply a distance function to item vectors obtained from a representation model, it is necessary to check if the vectors satisfy the requirements that are specified by the domain of that function. In the following, we revisit which pairs composed of a representation model and a distance function are compatible, pointing out how requirements are satisfied and if the intuition of the distance function is aligned with the representation model.

As seen earlier, Model C (document-term matrix) and Model U (user-item matrix) produce compositional item vectors, since vector components can be framed as the proportion of some whole and, thus, components only assume non-negative real values. Assuming that weighting and smoothing, if applied, did not introduce negative values, item vectors from these models are compatible with all distance functions there were presented:

- Both the Euclidean distance and the cosine distance require that $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, and the Jaccard distance requires $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n_+$. Thus, they can be directly applied to item vectors from document-term and user-item matrices, with or without smoothing.

- The Kullback-Leibler and the Jensen-Shannon divergences, as well as the Aitchison distance, require that $\mathbf{v}, \mathbf{w} \in \mathbb{S}^n$, so the item vectors from document-term and user-item matrices must be closed (Equation 26) and smoothed (Equation 27).

In turn, Model D (factorised document-term matrix) and Model V (factorised user-item matrix) produce non-compositional item vectors, since its components may assume negative real values; this fact restrains the set of compatible distance functions:

- The Euclidean and the cosine distances can properly handle item vectors with negative components, since they only require that $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$.

- The Kullback-Leibler and the Jensen-Shannon divergences, as well as the Jaccard and the Aitchison distances, cannot be safely applied to item vectors from these factorised models. Besides violating the intuition behind these distance functions, the presence of negative components may lead to an undefined value caused a division by zero or the logarithm of a negative number, which does not have a solution in $\mathbb{R}$.

Finally, it must be noted that the idea of item comparison is defined by a relationship between item vectors and a distance function: let i, j, k be item vectors from the model X; a distance function can be used to determine if j is more (dis)similar to i than k is.

2.7 The recommendation algorithms

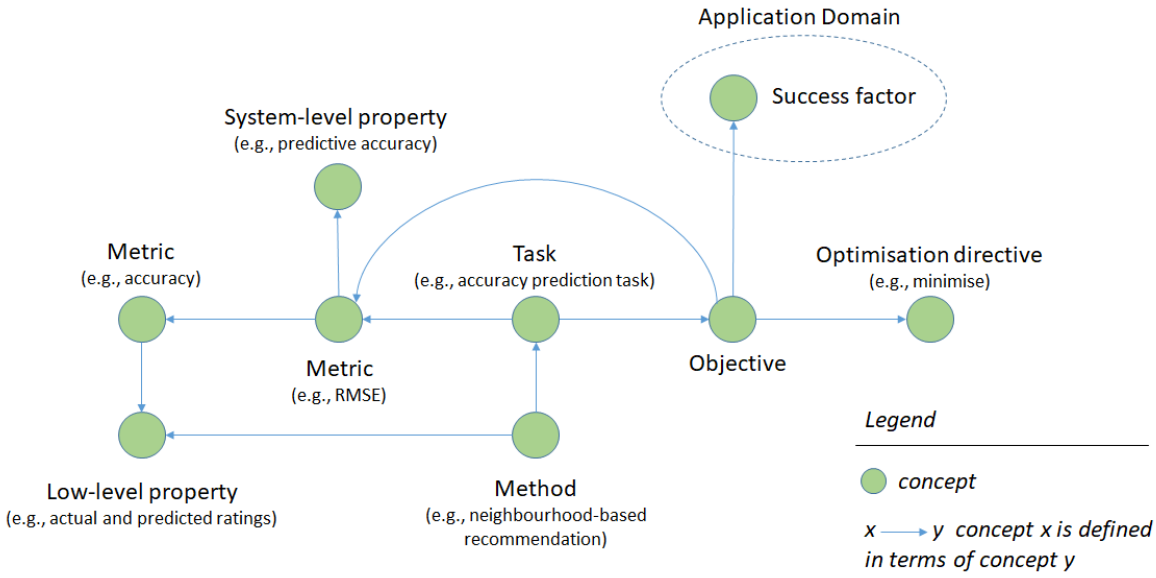
In the taxonomy presented in Section 2.1, it was mentioned that a recommender system usually keeps numerical representations of items so that its recommendation method can produce new recommendations. In this section, it is described how different recommendation methods handle item vectors to produce new recommendations lists. This section is divided in two segments: in Section 2.7.1, the concepts of recommendation task and recommendation method are defined, as well as their connections with the notion of objective and objective optimisation; and, in Section 2.7.2, the concept of recommendation algorithm is presented, and variants of the factorisation-based and neighbourhood-based recommendation algorithms are detailed and analysed.

2.7.1 Recommendation methods

The basic task performed by any recommender system is to produce recommendation lists to its users. We argue that the concept of "task" is an organising idea in recommender systems, as illustrated in Figure 5, and performs an integrative and a communicative role:

Integrative role: the notion of task can be used to integrate concepts in the application domain with design options in an RS. In this usage, a task describes a) a set of objectives, which are assumed to be linked to success factors in the application domain, and

Figure 5 – Conceptual links between a task and method in recommendation.



Source: Andre Paulino de Lima, 2019

b) how to assess progress towards these objectives. For example, suppose that the owner of a book recommending system is faced with the following challenge: the system must select, among the set of books unknown to a user u, the item that would obtain the highest rating from the user u. In this case, one may assume that, in the book recommendation domain, users are exclusively interested in books that were not previously read by them, and the capacity of predicting the rating that a user will give to a new recommended book is a key success factor. Alternatively put, this is an accuracy prediction task, which has a) just one objective: to predict the rating a user will give to a new item as accurately as possible, and b) a single metric: the difference between the predicted rating and the actual rating given by the user; the difference is expected to be as small as possible. It must be noted that the notion of objective connects an item-level property (predicted and actual item rating), an aggregative metric (e.g., RMSE) that is linked to a system-level property (predictive accuracy), and an optimisation directive (minimise).

Communicative role: the same concept can be used to organise research communities or industry practitioners around a well-defined goal. In the research community, it fosters comparability of results, since a task prescribes how to measure progress towards the objective. In the industry, it offers a common language that can be used by recommender system

specialists and domain experts to trace business goals to elements in the application design.

The notion of task can also link a set of objectives with a way to achieve them: a recommendation task is solved by a recommendation method. In this sense, a recommendation method is a "recipe" of how to select items for recommendation. In its elementary form, a recommendation method assumes that the selection of an item i should not directly depend on features of other items; more advanced methods may assume otherwise. A basic recommendation method is detailed in Algorithm 1, and its steps are described as follows:

Input parameters

1. Parameter $\mathtt{u}$: the user to which a recommendation list is being produced.
2. Parameter $\mathtt{L}_1$: a list of candidate items, usually consisting of a random sample of the items that are unknown to the target user.
3. Parameter $\mathtt{topN}$: indicates the number of items recommendation list must contain.
4. Parameter Θ: the user preference model, which takes $\mathtt{u}$ and an item i as parameters, and produces a score that is proportional to the rating the user $\mathtt{u}$ would give to i.

Procedure

- Line 2: each item i in the list of candidate items $\mathtt{L}_1$ is assessed by means of the model Θ, and a score $\hat{r}_{ui}$ is produced. A new list L_2 is created from the pairs $(i, \hat{r}_{ui})$.
- Line 3: items in L_2 are sorted by decreasing order of score, creating a new list L_3.
- Line 4: the first $\mathtt{topN}$ items from L_3 are selected as the recommendation list L_4. Alternatively, if $\mathtt{topN} = 1$, the list is composed of a single recommended item.

Algorithm 1 A basic recommendation method that receives a list of candidate items.

1: **procedure** BASIC-RECOMMENDATION-METHOD($\mathtt{u}, \mathtt{L}_1, \mathtt{topN}, \Theta$)
2: $\quad L_2 \leftarrow map(\, i \in \mathtt{L}_1, (\, i, \hat{r}_{ui} \leftarrow \Theta(\mathtt{u}, i))\,)$
3: $\quad L_3 \leftarrow sort(L_2, key = -\hat{r}_{ui})$
4: $\quad L_4 \leftarrow map(\, (i, \hat{r}_{ui}) \in L_3[1 : \mathtt{topN}],\, i)$
5: $\quad$ **return** L_4

Source: Andre Paulino de Lima, 2019

2.7.2 Recommendation algorithms

In the taxonomy presented in Section 2.1, it was shown that systems in each category produce new recommendations from different information (ratings, content, or a combination thereof). This difference is usually accommodated by the user preference model, which is an abstraction of the relationship between the user and its preference for items. Two examples: a) in CFRS, a new recommendation is produced from the ratings given by the whole user community; thus, the preference model can be induced from the entire user-item matrix $\mathbf{R}$; b) in CBRS, a new recommendation is produced from the ratings given by the target user, but item content can also be used; thus, the preference model can be induced from the entire document-term matrix $\mathbf{D}$, plus the user ratings vector $\mathbf{r}_u$. When details of how the user preference model is induced and applied to item vectors are made explicit (and unambiguous), we have a recommendation algorithm.

There are two main approaches that can be followed to induce a preference model: the neighbourhood-based and the factorisation-based approaches. In the neighbourhood-based approach, the preference model usually obtains the score for an item i by solving a weighted k-nearest neighbours interpolation. As will soon be detailed, the item-kNN recommendation algorithm (KOREN; BELL, 2015) employs a model that follows this approach. On the other hand, in the factorisation-based approach, the preference model generally obtains the score for an item i by inner-product reconstruction. The PureSVD recommendation algorithm (CREMONESI; KOREN; TURRIN, 2010), and the FunkSVD algorithm (KOREN; BELL, 2015) employ models that follow this approach.

Despite the differences between the neighbourhood and the factorisation approaches, there is an interesting result that relates them: under certain conditions, in so far as the size of the neighbourhood approaches the number of items in the system, a recommendation algorithm equipped with a neighbourhood-based model becomes increasingly equivalent to the same algorithm using a factorisation-based model (KOREN; BELL, 2015). The idea that there is a graded variation that has the neighbourhood and factorisation approaches as its extremes inspired the selection of preference models that are presented in this section. This inspiration is not in the sense of exploring the effect of varying a neighbourhood size parameter. Instead, as one of our research questions inquires into the effect on surprise of item representation and item comparison, we seek to explore a graded variation of

item representation and item comparison that can affect the surprise metric as much as the preference model. To make this graded variation explicit, the preference models are presented in a progressive order of dependence, which is summarised in Table 2: the choices of item representation and item comparison adopted by these preference models move from being independent of those employed by the surprise metric (algorithm FP) towards using the same definitions used to assess surprise (algorithm N3).

PureSVD, a factorisation-based recommendation algorithm

The PureSVD algorithm (FP) corresponds to the Algorithm 1 with a preference model Θ that uses an inner product reconstruction to produce the score for an item i:

$$\Theta(u, i) = \mathbf{r}_u \cdot \mathbf{Q} \cdot \mathbf{q}_i^T = \mathbf{p}_u \cdot \mathbf{q}_i^T \,, \tag{33}$$

with $\mathbf{r}_u$ as the user vector (u^{th} row of the user-item matrix $\mathbf{R}$), and the matrix $\mathbf{Q}$, which contains the parameters of the preference model, is obtained from the entire (sparse) matrix $\mathbf{R}$ by applying truncated SVD to obtain the best rank f approximation to $\mathbf{R}$[15]:

$$\mathbf{R}_{(m,n)} \approx \mathbf{U}_{(m,f)}\, \mathbf{\Sigma}_{(f,f)}\, \mathbf{Q}^T_{(f,n)}.$$

This transformation can be interpreted as the mapping of the user and item vectors from the matrix $\mathbf{R}$ to their counterparts in a latent factor space. In other words, a n-dimensional

[15] It should be noted that the matrix $\mathbf{Q}$ obtained by this procedure corresponds to the same matrix $\mathbf{Q}$ on which the representation Model V is based, as was seen in Section 2.5.2.

Table 2 – Combinations of item representation and item comparison explored by the recommendation algorithms. They gradually move from adopting item representation and comparison schemes that are independent of those used to assess surprise (FP) towards adopting the same item representation and comparison that are used by the surprise metric (N3).

Alg.	Item representation	Item comparison
FP	Item vectors from the $\mathbf{Q}$ matrix (Model V), which may differ from the item representation adopted to assess surprise.	Does not properly perform item comparison; instead, it estimates the user-item interaction by inner product reconstruction ($\hat{r}_{ui} = \mathbf{r}_u \cdot \mathbf{Q} \cdot \mathbf{q}_i^T$).
N1	Item vectors from the $\mathbf{Q}$ matrix (Model V), which may differ from the item representation adopted to assess surprise.	Uses the **same** distance function employed by the surprise metric.
N2	Uses the **same** item representation adopted by the surprise metric.	Uses cosine distance, which may differ from the distance employed to assess surprise.
N3	Uses the **same** item representation adopted by the surprise metric.	Uses the **same** distance function employed by the surprise metric.

Source: Andre Paulino de Lima, 2019

user vector $\mathbf{r}_u$, which corresponds to a row vector of the matrix $\mathbf{R}$, is mapped to a f-dimensional user vector $\mathbf{p}_u$, whereas a m-dimensional item vector $\mathbf{r}_i$, which corresponds to a column vector of $\mathbf{R}$, is mapped to a f-dimensional item vector $\mathbf{q}_i$; both $\mathbf{p}_u$ and $\mathbf{q}_i$ belong to the same latent factor space, but the former captures how the user u appraises each latent factor, and the latter captures the presence of each latent factor in the item i (CREMONESI; KOREN; TURRIN, 2010). As this preference model is induced exclusively from community ratings, it can be used to equip recommendation algorithms for CFRS.

In regard to item representation and comparison, it must be noted that the preference model requires specific vector representations: the user vector $\mathbf{r}_u$ and the matrix $\mathbf{Q}$. Moreover, it does not need a distance function to assess (dis)similarity between items. This means that, in a recommender system using PureSVD, the choices of item representation and distance function used in the assessment of surprise can be changed, but these changes cannot be extended to the preference model and, as a result, cannot be extended to the recommendation algorithm.

Item-kNN, a neighbourhood-based recommendation algorithm

The item-kNN algorithm corresponds to the Algorithm 1 with a model Θ that solves a weighted k-nearest neighbours interpolation to obtain the score for an item i:

$$\Theta(u,i) = b_{ui} + \frac{\sum\limits_{j \in S^k(i,u)} s_{ij}\,(r_{uj} - b_{uj})}{\sum\limits_{j \in S^k(i,u)} s_{ij}}, \tag{34}$$

with r_{uj} as the rating that the user u has given to the item j; the matrices $\mathbf{B}$ and $\mathbf{S}$, which hold parameters of the preference model, are obtained from the user-item matrix $\mathbf{R}$, and possibly from the document-term matrix $\mathbf{D}$. The matrix $\mathbf{B}$ is invariably induced from the matrix $\mathbf{R}$, and corresponds to the baseline predictor $b_{ui} = \mu + b_i + b_u$, with:

$$\mu = \frac{1}{C} \sum_{u,i|r_{ui}\neq 0} r_{ui};\;\; b_i = \frac{1}{C_i} \sum_{u \in U|r_{ui}\neq 0} (r_{ui} - \mu);\;\; b_u = \frac{1}{C_u} \sum_{i \in I|r_{ui}\neq 0} (r_{ui} - \mu).$$

with μ as the global average rating, b_u as the normalised average rating given by the user u, and b_i as the normalised average rating given to the item i. On the other hand, the matrix $\mathbf{S}$ corresponds to a weighted similarity matrix:

$$s_{ij} = \frac{n_{ij} - 1}{n_{ij} - 1 + \lambda}\,\rho_{ij},$$

with $\mathbf{N} = (n_{ij}) \in \mathbb{N}^{n \times n}$ as the number of users that rated both items i and j (n_{ij} is obtained from $\mathbf{R}$), and λ as a regularising parameter (e.g. $\lambda = 100$). The matrix $\boldsymbol{\rho}$ is a similarity matrix, which is derived by the element-wise application of Equation 15 or 16 to a distance matrix $\mathbf{M}$ such that $m_{ij} = dist(i, j)$. Finally, the predicate $S^k(i, u)$ uses the matrix $\boldsymbol{\rho}$ to compute the set of the k-nearest neighbours of the item i that have been rated by the user u. It should be noted that the distance matrix $\mathbf{M}$ depends on choices of item representation and distance function. Therefore, it can be created from item vectors extracted from any of the representation models described in Section 2.5. As a result, the matrix $\mathbf{S}$ can be computed from content or rating data. Since the matrix $\mathbf{B}$ is invariably obtained from rating data, this preference model can be induced a) exclusively from rating data, or b) from a combination of rating and content data. In the former, the model can equip algorithms for CFRS, and in the latter, the model can equip algorithms for CBRS.

In regard to item representation and comparison, it must be noted that this preference model is more flexible than the one used in PureSVD because it can handle item vectors from any model. Moreover, it requires a distance function to assess (dis)similarity between items. This means that, in a recommender using an item-kNN algorithm, the choices of item representation and distance function used in the assessment of surprise can be changed, and these changes can be extended to the preference model; as a result, the changes will also affect the recommendation algorithm, unlike is the case with the PureSVD. To explore the idea of a graded variation, we describe three variants of the item-kNN algorithm, moving from the choices of item representation and comparison adopted by the PureSVD towards the similar choices adopted by the surprise metric:

- *Item-kNN algorithm with shared distance (N1)*: the preference model adopts the distance function d that is used by the surprise metric, but employs the item representation that is used by the PureSVD algorithm. This means that N1 uses the preference model defined in Equation 34 combined with a distance matrix $\mathbf{M}$ that is obtained by applying the distance function d to item vectors from the matrix $\mathbf{Q}$.

- *Item-kNN algorithm with shared representation (N2)*: the preference model adopts the same representation model (X) that is used in surprise assessment, but invariably employs the cosine distance. This means that N2 uses the preference model in Equation 34 with a distance matrix $\mathbf{M}$ that is obtained by the pairwise application of the cosine

distance to item vectors from the representation model X. The cosine distance was selected because, as seen in Section 2.6, it consists of an affine transformation of the inner product between two vectors normalised to unit length, and we wanted this distance function to gradually depart from the inner product, which is used in PureSVD.

- *Item-kNN algorithm with shared config (N3)*: uses the preference model in Equation 34 combined with the same distance function d and representation model X that are employed by the surprise metric. The matrix $\mathbf{M}$ is obtained by applying d to item vectors from model X.

In summary, the recommendation algorithms presented a) can be applied in CFRS (FP and N1; N2 and N3, with some restrictions) and in CBRS (N2 and N3, with some restrictions), b) cover factorisation-based (FP) and neighbourhood-based (N1, N2, N3) approaches, and c) can explore multiple choices of item representation and item comparison.

2.8 The evaluation methods

All the metrics that were described in the Section 2.3 evaluate the surprise of a single recommendation list that was produced by the system that is being evaluated. To obtain an estimate of how the system performs with regard to surprise, an evaluation method is required. Most studies adopt an evaluation method that implements a statistical procedure to compute this kind of system-level estimate: a sample of users is selected, recommendation lists are produced, surprise evaluations are made, and the average is calculated. However, there are other evaluation methods that can be used for this purpose, depending on the experiment design adopted by the research. In this section, we describe a taxonomy of experiment designs commonly used in RS research and justify the adoption of an off-line experiment by this project (Section 2.8.1), and then focus on describing a state-of-the-art evaluation method for surprise known as one plus random (Section 2.8.2).

2.8.1 A brief taxonomy of experiment designs

When selecting which experiment design is the most appropriate to a research project, multiple factors are taken into account, including the nature of the research question and the resources that are available to conduct the research. In fact, Gunawardana and Shani (2015) present a taxonomy of experiment designs frequently adopted in research related to recommender systems. This taxonomy divides experiment designs into three broad categories: a) off-line experiments; b) user studies; and c) online evaluation:

a) Off-line experiment: in this experiment design, the evaluation of an RS is conducted without the participation of real users. Instead, it assumes that a dataset containing interactions of real users with an RS is available. In general, these interactions comprise the ratings that users attributed to items over a period of time, possibly having a temporal reference of when each rating was created. The dataset may also contain data that describe the items in the repository of the RS. The type of data that can be collected from an off-line experiment is limited to quantitative measurements computed over ratings and, when available, on content that is combined with the items. This means that collecting qualitative measurements, which may be more informative about subjective aspects of the user interaction with the system, is beyond the scope of this type of experiment, as much as measurements aimed at evaluating the effect on user behaviour or perception of a change in system parameters. Therefore, this experiment design is appropriate if the research question can be approached by analysing quantitative measurements derived from the dataset. In general, this is the first experiment design to be considered because it is the simplest and the cheapest one to conduct: there is no need to recruit users and, most of the times, a public dataset widely employed by the research community can be used.

b) User studies: this experiment design assumes that a representative group of real users will participate in the experiment. This experiment design usually consists of allocating each user to a set of predefined tasks. Each task demands that the user interact with the RS to achieve a specific purpose, and must be carefully designed to produce the data required to assess some aspect of the research question. Both quantitative and qualitative measurements can be collected from the experiment, as

well as correlations between these measurements. It must also be noted that this experiment design allows for the comparative evaluation of multiple instances of RS in a single experiment. However, since the interaction of users with the RS is limited to the duration of the experiment, this design is not indicated to assess the effect on user behaviour or perception of a change in system parameters.

c) Online experiments: this experiment design assumes that a group of real users will take part in the experiment, and also assumes that the RS under evaluation is a production instance. In general, the production instance is adapted to redirect the traffic of an uninformed, random sample of the users to a condition that explores some aspect of the research question. For example, if the research question requires the assessment of the impact of changing the current recommending algorithm A to a new recommending algorithm B, the sample of users allocated to the condition will receive recommendations produced by algorithm B, while the remainder of the users will still be served with recommendations produced by the algorithm A. Quantitative measurements can be collected during the online evaluation, but collecting qualitative measurements is more limited, if compared to user studies, owing to the fact that the users in the experiment were not informed of their participation; thus, any qualitative assessment that requires the contextualisation of the participants is hampered.

In view of this, an off-line experiment was selected as the fundamental experiment design for this research project because:

- As will be clarified in Chapter 3, the research questions involve estimating the potential surprise of an RS, which is a quantitative measurement.
- There are public datasets widely used by the research community that have been employed in research projects involving surprise assessment.
- The research questions do not aim to determine if the current intuitions on surprise correspond to the surprise experienced by real users when interacting with a system, nor do they aim to assess effect of changes in system parameters on user perception. Instead, they seek to investigate the effect on system-level surprise of distinct, but plausible, models of surprise, represented by different forms of the surprise metric.

2.8.2 Off-line evaluation methods

As mentioned earlier, an off-line experiment assumes that a dataset containing user ratings (and possibly other data) is available, and that this data can be used to obtain quantitative measurements. In general, to assess a research hypothesis, the individual measurements must be aggregated, which is a task usually performed by an evaluation method: it orderly transforms the individual measurements, as defined in the experiment design, and aggregate them as required by the selected hypothesis significance test.

A typical evaluation method for off-line experiments adopts an approach that is similar to the one employed in the machine learning research: the dataset is segmented into two parts: a training partition and a test partition. The training partition is employed to induce (or train) models that capture particular aspects of the RS (e.g. user preference models) and the test partition is used to evaluate how well the models represent the aspect of interest (e.g. how accurately the user preference model predicts the rating that an arbitrary user would give to an item in the test partition).

The one plus random evaluation method

An off-line evaluation method has been proposed to evaluate systems that operate in domains in which they need to solve a top-N recommendation task, which has a) at least one objective: to select the N most promising items to an arbitrary user, and b) at least one metric: recall, which is explained ahead. This method, known as one plus random (KOREN, 2008; CREMONESI et al., 2008; BELLOGIN; CASTELLS; CANTADOR, 2011), follows the intuition that, in a sufficiently large set L_1 that consists of items unknown to the user u, most of these items are irrelevant to u. Given an item i^* that has been highly rated by u, the method defines that if a recommendation algorithm attributes to i^* a score such that i^* ranks among the top N items of $L_1 \cup \{i^*\}$, then the algorithm has succeeded in the task. This evaluation method is being presented here because the state-of-the-art off-line evaluation method for surprise, as we will soon see, was inspired in this method. The one plus random method is detailed in Algorithm 2, and its steps can be described as follows:

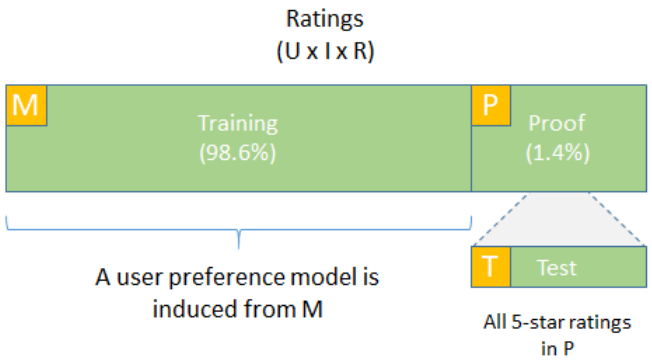
Input parameters

1. Parameters `M` and `T`: the dataset, composed of ratings in the format (u, i, r_{ui}), must be split into two parts: a training set and a proof set, referred to as set `M` and set `P`, respectively. In fact, Cremonesi, Koren and Turrin (2010) suggest that 98.6% of the samples be allocated in `M` and the remainder in `P`, following the split used in the Netflix competition, as illustrated in Figure 6. The allocation ratio between `M` and `P` is intended to improve the comparability of the results obtained by different studies. All samples that were allocated to the set `P` and that correspond to the highest possible rating must be also allocated to the test set, referred to as set `T`. In the Netflix dataset, the ratings range within the discrete scale of 1-5 stars and, therefore, only the 5-star ratings were allocated to the set `T`.

2. Parameter `topN`: indicates the number of items a recommendation list must contain.

3. Parameter `metamodel`: an external procedure that can be used to induce a user preference model from the ratings in the training set `M` (e.g. PureSVD, item-kNN, neural networks) and possibly from content data. As seen in the taxonomy presented earlier, systems from different categories use distinct approaches to produce new recommendations, and this fact influences the selection of the `metamodel`.

Figure 6 – Segmentation in the Netflix prize: the dataset was split into partitions M and P, and the subset of P with the highest ratings was assigned to partition T.



Source: Andre Paulino de Lima, 2019

Procedure

- Line 3: a user preference model is induced from the training set `M`. The `metamodel` produces the required model parameters, as seen in Section 2.7.2: if the `metamodel` must create a PureSVD model, then the matrices $\mathbf{U}, \mathbf{\Sigma},$ and $\mathbf{Q}$ are computed; if the `metamodel` creates an item-kNN model, then the matrices $\mathbf{B}$ and $\mathbf{S}$ are computed. Instead of the whole $\mathbf{R}$ matrix, only its subset represented by the training set `M` is employed. For the sake of generality, matrices that can be computed from content data (e.g. matrix $\mathbf{M}$) are assumed to be available in the environment. And, for the sake of clarity, the notation used in the algorithm assumes that a generic model Θ is induced, and the estimated score for an item i by the user u is generically expressed as $\hat{r}_{ui} \leftarrow \Theta(i, u)$.
- Line 4: the variable u is assigned to each distinct test user. The predicate $U(\mathtt{T})$ returns the set of users that have some rating in the test set `T`. The order in which users in $U(\mathtt{T})$ are assigned to u is arbitrary.
- Line 5: the variable i^* is assigned to each item in the test set `T`. The predicate $I(\mathtt{T}, \{u\})$ returns the set of items in `T` that have been rated by the user u. The order in which items in $I(\mathtt{T}, \{u\})$ are assigned to i^* is arbitrary. As mentioned earlier, all items in the test set have been rated with the highest possible grade.
- Line 6: The predicate $I(\mathtt{M}, U(\mathtt{M}))$ returns the set of items in the training set `M`, and $I(\mathtt{M}, \{u\})$ returns the set of items in `M` that have been rated by the user u. Thus, L_0 contains all the items in the training set `M` that are unknown to the user u.
- Line 7: the variable L_1 is assigned to a set containing 1,000 items randomly selected from L_0 plus the item i^*.
- Line 8: the variable L_4 is assigned with a list containing the `topN` items selected by the recommendation algorithm, which was described in Algorithm 1.
- Line 9: the number of hits is increased if the item i^* is one of the `topN` items in L_4.
- Line 10: in the end, the system-level recall is computed as the number of *hits* divided by the number of samples that were allocated to the test set `T`.

Algorithm 2 One plus random method applied to estimate the recall of an RS

```
1: procedure ONE-PLUS-RANDOM-RECALL(M, T, topN, metamodel)
2:     hits ← 0
3:     Θ ← metamodel(M)
4:     ∀u ∈ U(T) :
5:         ∀i* ∈ I(T, {u})
6:             L_0 ← I(M, U(M))\I(M, {u})
7:             L_1 ← random( L_0, 1000 ) ∪ {i*}
8:             L_4 ← basic-recommendation-method(u, L_1, topN, Θ)
9:             hits ← hits + 1(i* ∈ L_4)
10:    recall ← hits/|T|
11:    return recall
```

Source: Andre Paulino de Lima, 2019

The one plus random evaluation method adapted to surprise

In a recent study, Kaminskas and Bridge (2014) adapted the one plus random method to estimate the degree of surprise of recommender system. The original intuition behind the one plus random method is retained, but there are two relevant modifications:

- In its original form, all items in the set T are assessed (Line 5), while only one item per user is assessed in the adapted form. This modification grants two benefits. The first one is that the method requires fewer computing resources since users with a large number of ratings in the set T will be represented by a single item. The second benefit refers to representativeness of each test user. For example, suppose that user u_1 has 10 ratings allocated in the test set T, and user u_2 have only one rating in T. In the final result, both users will have equally contributed to the measurement.

- The assessment of surprise is included in the procedure, so the method produces an estimate for both recall and surprise.

The one plus random method adapted to estimate surprise of a recommender system is detailed in Algorithm 3; its input parameters are the same ones that have been described in the original one plus random method, and its steps can be described as follows:

Procedure

- Line 3: a user preference model is induced from data allocated to the training set M.

- Line 4: the variable u is assigned to each test user.
- Line 5: the variable i^* is assigned to a random item in `T` that has been rated by u.
- Line 6: the variable E_u is assigned to the set of items in the dataset ($\mathtt{M} \cup \mathtt{P}$) that have been rated by the user u.
- Line 7: the variable L_0 is assigned with the set of items in `M` that are unknown to u.
- Line 8: the variable L_1 is assigned to a set containing 1,000 items randomly selected from L_0 plus the item i^*.
- Line 9: the variable L_4 is assigned with a list containing the `topN` items selected by the recommendation algorithm, which was described in Algorithm 1.
- Line 10: each element i of the recommendation list L_4 is mapped to its surprise $S_i(i, E_u)$. The predicate S_i was described in Section 2.3.6, Equations 11 and 12. Thus, the variable L_5 is assigned with a list of the surprise obtained from each item in L_4.
- Line 11: the number of hits is increased if the item i^* is one of the `topN` items in L_4.
- Line 12: the surprise of the recommendation list L_4 is accumulated.
- Line 13: the system-level recall is computed as the number of *hits* divided by the number of users in the test set `T`.
- Line 14: the system-level surprise is computed as the surprise of each recommendation list generated during the evaluation divided by the number of test users times the list length, `topN`. This value represents the average surprise per item obtained from a sample of users and ranges in the interval defined by the S_i metric.

Algorithm 3 One plus random method applied to estimate the surprise of an RS

1: **procedure** ONE-PLUS-RANDOM-SURPRISE(`M`, `P`, `T`, `topN`, `metamodel`)
2: $hits \leftarrow 0$; $acc \leftarrow 0$
3: $\Theta \leftarrow \mathtt{metamodel}(\mathtt{M})$
4: $\forall u \in U(\mathtt{T})$:
5: $i^* \leftarrow random(I(\mathtt{T}, \{u\}), 1)$
6: $E_u \leftarrow I(\mathtt{M} \cup \mathtt{P}, \{u\})$
7: $L_0 \leftarrow I(\mathtt{M}, U(\mathtt{M})) \backslash I(\mathtt{M}, \{u\})$
8: $L_1 \leftarrow random(\, L_0,\, 1000\,) \cup \{i^*\}$
9: $L_4 \leftarrow$ *basic-recommendation-method*$(u, L_1, \mathtt{topN}, \Theta)$
10: $L_5 \leftarrow map(\, i \in L_4,\, S_i(\, i, E_u)\,)$
11: $hits \leftarrow hits + 1(i^* \in L_4)$
12: $acc \leftarrow acc + sum(L_5)$
13: $recall \leftarrow hits / |U(\mathtt{T})|$
14: $surprise \leftarrow acc \,/\, (\mathtt{topN} \times |U(\mathtt{T})|)$
15: **return** $(recall, surprise)$

Source: Andre Paulino de Lima, 2019

It should be noted that this method is being presented because it inspired the evaluation method to assess surprise that will be proposed in the next chapter.

2.9 Closing remarks

This chapter began by presenting a taxonomy that classifies recommendation systems according to the information that is used by their recommendation algorithm to produce new recommendations. This taxonomy was used as a unifying, guiding theme in a bottom-up narrative: a) we first showed that items must be represented according to the requirements of the category of the recommender (e.g. CFRS, CBRS), which translates requirements of a recommendation algorithm; b) that it is often the case that items must be compared by the recommendation algorithm when producing a new recommendation; and finally c) that recommendation algorithms are fundamentally defined by how the ideas of item representation and item comparison are articulated. These ideas are also fundamental to the definition of surprise, as was presented in the review of several metrics in the literature of recommender systems. We sought for insights from research in cognitive models of surprise to assess the plausibility of the metrics, aiming to combine competing ideas on how to assess the effect on system-level surprise of this or that surprise model, supposing they are faithful and adequately correspond to reality. As will be seen in the next chapter, the idea of potential surprise is fundamentally based on this analysis.

3 Foreground

In this chapter, the primary objective of this project, which is the development of a metric for surprise that takes account of the growth in user experience and in the size of the repository, is fulfilled. To achieve this goal, we address the first three research questions, which inquire into the limits to surprise in different scenarios:

RQ1 Are there limits to how much surprise a recommender can offer to a user?
RQ2 Are there limits to how much surprise can be embedded in a recommendation?
RQ3 If these limits exist, is it possible to use them to create a measurement scale up in which the performance of a system can be measured?

Our main contribution, which is an evaluation method that factors into the system-level surprise the inflation of the set of items, is also developed. The remainder of this chapter is organised as follows: in Section 3.1, it is argued that a metric for surprise of RS should have some properties that are imposed by a user-centred view of surprise. This is important to justify the adoption of the definition of surprise proposed by Kaminskas and Bridge (2014) as a reference metric to estimate the surprise of an item. The behaviour of this surprise metric is analysed in Section 3.2 with regard to some scenarios that are used to support the theoretical model. In Section 3.3 a theoretical model of surprise is developed to answer the research questions RQ1 and RQ2; as a result, a new metric for surprise of RS called "normalised surprise of a sequence" is defined. Finally, in Section 3.5, the off-line evaluation method for surprise described in Section 2.8.2 is adapted to apply the newly defined surprise metric, which addresses RQ3.

3.1 Revisiting the property of surprise

As described in Section 2.3, there are several metrics of surprise in the literature on RS. They explore diverse intuitions on how to assess the surprise experienced by a user when exposed to a recommendation list. However, as argued in Section 2.4, a metric for surprise of recommender system would benefit from and should account for the ideas that have been explored in the cognitive science about how humans experience surprise, such as a) surprise is subjective in nature, b) surprise of an unobserved item changes as a user is exposed to new experiences, and c) surprise requires assessing discrepancy (or distance)

between what is expected and what is observed.

Surprise must be subjective: the studies on cognitive models of surprise that were cited in Section 2.4 portray surprise as being subjective in nature, and show that its quantitative models usually involve some form of subjective probability, which may represent expectations or beliefs held by an individual that result from their past experience. From a more intuitive standpoint, and when applied to recommender systems, this idea can be illustrated through the following scenario:

Scenario 3.1 *Two users u_1 and u_2 have been exposed to the same set of items, E. Initial conditions:*

- *there are two items, namely i and j, that are not in E;*
- *items i and j are very dissimilar to any item in E;*
- *items i and j are very similar to each other.*

Under these conditions:

- *If the system recommended item i to user u_1 or user u_2, it would be a surprising recommendation, because the item i is very dissimilar to any item in E.*

Now assume that, in addition to the initial conditions, the user u_2 has been exposed to item j. Then, under this new condition:

- *If the system recommended item i to user u_1, it would be a surprising recommendation, because the item i is very dissimilar to any item in E;*
- *If the system recommended item i to user u_2, it would not be as surprising, because now the user u_2 knows j, which is similar to item i.*

The event of "recommending item i to user u_1" may produce a different outcome (in terms of surprise) than that of "recommending item i to user u_2", because the users may have been exposed to different sets of items and, thus, have a different past experience. ■

Surprise must be dynamic: assuming that surprise depends on beliefs or expectations, it seems reasonable to presume that it changes over time, as the user is constantly being exposed to new experiences. This idea can be illustrated through the following scenario:

Scenario 3.2 *User u_1 has been exposed to the set of items E.*
Initial condition:

- *there are two items, namely i and j, that are not in E;*
- *items i and j are very dissimilar to any item in E;*
- *items i and j are very similar to each other.*

Under this condition:

- *If the system recommended item i to user u_1, it would be a surprising recommendation, because the item i is very dissimilar to any item in E.*

Now assume that user u_1 has been exposed to item j. Then, under this new condition:

- *If the system recommended item i to user u_1, it would not be as surprising, because now the user u_1 knows j, which is similar to item i.*

The same event, the recommendation of item i to user u_1, have different outcomes in terms of surprise, depending on the past experience of the user u_1. ■

Surprise is related to the notion of distance: all the metrics reviewed in Section 2.3 involve the notion of distance; most of them reflect the extent to which a new item resembles the items known to a user. This is in accordance with the subjective and dynamic views of surprise, since both involve assessing similarity between objects.

In adopting these three ideas as premises for this work, and taking account of the analysis of the coherence between surprise metrics and the two cognitive models of surprise briefly discussed in Section 2.4, we support the structure of a surprise metric defined by Kaminskas and Bridge (2014), which is described in Equation 11. This definition of surprise of an item, $S_i(i, E_u)$, a) portrays surprise as being proportional to the degree to which the item i is dissimilar to the items known to the user; b) adopts a subjective view, since it defines surprise as a function of the items known to the user; and c) also accounts for changes in the surprise of an unobserved item as the growth of the set of known items. It must be noted that we support the structure of S_i, and not its full definition, because we chose to keep open the option of distance function for further experimental exploration.

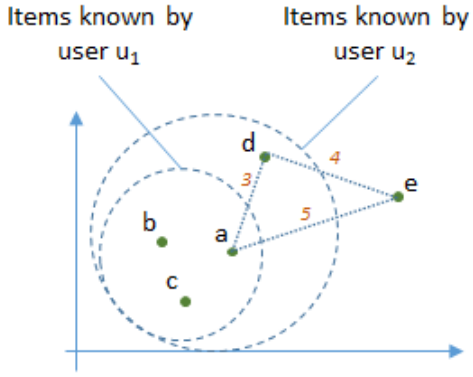
3.2 The behaviour of the S_i metric

A closer analysis of the S_i metric reveals three important results that must be made clearer for they are fundamental to the theoretical model that will be developed in the next section. The first two results have already been intuitively explored in the previous section, and are presented now with a numerical example exploring the Euclidean distance, and the third is an extension of the idea of surprise as being dynamic in nature:

a) The surprise of an item is not necessarily the same for every user (Scenario 3.1);
b) The surprise of an item changes over the time (Scenario 3.2);
c) The surprise of a sequence of items is order dependent.

To develop these results, the fictitious recommender system that is illustrated in Figure 7 is employed: it has two users, u_1 and u_2, and five items, namely $I = \{a, b, c, d, e\}$. Each item is represented as a point in the $\mathbb{R}^2$ plane. The figure illustrates the initial state, in which the user u_1 has been exposed to items $E_{u_1} = \{a, b, c\}$ and the user u_2 has been exposed to items $E_{u_2} = \{a, b, c, d\}$. Moreover, the distances between items a, b and c are annotated in the diagram: $d(a, d) = 3$, $d(a, e) = 5$ and $d(d, e) = 4$. It must be noted that the item a is more similar (is nearer) to items d and e than are the items b and c.

Figure 7 – A fictitious recommender system with two users and five items.



Source: Andre Paulino de Lima, 2019

a) The surprise of an item is not necessarily the same for every user: as users u_1 and u_2 have different past experiences, which are respectively represented as E_{u_1} and E_{u_2},

the surprise of an item may vary. For example, the surprise of the item e to user u_1 is 5 and to user u_2 is 4, according to the S_i metric[1]:

$$S_i(e, E_{u_1}) = \min_{j \in E_u} dist(e, j) = dist(e, a) = 5,$$

$$S_i(e, E_{u_2}) = \min_{j \in E_u} dist(e, j) = dist(e, d) = 4.$$

b) The surprise of an item changes over the time: as a user u is exposed to new items, the size of the set E_u increases, reflecting growth in experience. If the surprise of an unobserved item[2] is assessed after each time the user is exposed to a new item, then its surprise will monotonically decrease. For example, in the initial condition illustrated in Figure 7, in assessing the surprise of the item e to user u_1, one would obtain the value 5. Now suppose that user u_1 has been exposed to item d. Then, under this new condition, in assessing the surprise of the item e to user u_1, one would obtain the value 4. The decrease is monotonic because, according to S_i, the user never "forgets" any items, and inserting new items in E_u can only reduce the current minimal distance between the unobserved item and the items in E_u.

c) The surprise of a sequence of items is order dependent: in other words, the total amount of surprise of a sequence of items depends on the order in which these items are presented. For example, suppose that the system recommends to user u_1 the item d, which is promptly consumed, and then the item e, which is also consumed. In this case, the total amount of surprise obtained is:

$$S_i(d, E_{u_1}) + S_i(e, E_{u_1} \cup \{d\}) = S_i(d, \{a, b, c\}) + S_i(e, \{a, b, c, d\}) = 3 + 4 = 7.$$

The term $E_{u_1} \cup \{d\}$ accounts for the fact that the user past experience increased after they were exposed to item d. If the order of the recommendations had been the opposite, the result would be different:

$$S_i(e, E_{u_1}) + S_i(d, E_{u_1} \cup \{e\}) = S_i(e, \{a, b, c\}) + S_i(d, \{a, b, c, e\}) = 5 + 3 = 8.$$

3.3 A theoretical model of potential surprise

An intuition that is fundamental to this work is the link between surprise and information: qualitative models of surprise in cognitive science hold that surprise emerges

[1] It must be noted that it is not possible to assess the surprise of an item to a cold-start user u, because in this case, $E_u = \varnothing$ and the S_i metric is undefined in this situation.

[2] In this sense, an unobserved item z is the last item that the system would recommend to the user.

when the subject (user) is faced with new information (ITTI; BALDI, 2009; FOSTER; KEANE, 2015; REISENZEIN; HORSTMANN; SCHÜTZWOHL, 2017). In view of this, and applying the intuition to recommender systems, a recommendation can be framed as a vehicle that conveys new information to the user. The limits to the capacity of a recommender system at embedding new information in its recommendations are central concerns of this work.

In brief, the main concepts are developed in this order: first, we approach the problem of how to compute the exact total amount of surprise an RS can offer to an arbitrary user (Sections 3.3.1 to 3.3.3); second, how to calculate an approximation to the total amount of surprise an RS can offer to an arbitrary user (Section 3.3.4), and finally, how to compute the exact and the approximate total amount of surprise an RS can embed in a recommendation list of length k (Section 3.3.5). To develop these concepts, the theoretical model assumes the existence of a computational simulation process that can be executed in an arbitrarily short period of time, as described in the following scenario:

Scenario 3.3 *A process simulating the exposure of all users to all items in the repository. Initial condition:*

1. *The recommender system has a finite number of items in its repository;*
2. *The repository of the system remains stable during the simulation;*
3. *Each user has rated at least one item;*
4. *The surprise that would result from recommending the item i to user u can be adequately estimated by the $S_i(i, E_u)$ metric described in Equation 11.*

Interactions:

5. *At each interaction, the system produces a recommendation list L for the user u that contains k items unknown to them;*
6. *Upon receiving a recommendation list L, an automated agent that serves as the proxy for the user u promptly consumes each item in the order they are ranked in L, and produces an estimate for the surprise that the user u would experience.*

Final condition:

- *After a finite number of interactions, all the user agents will have been exposed to all items in the repository, and the total amount of surprise available to each user and the total amount of surprise in the system have been estimated.* ■

Some conditions in this scenario are arguably incompatible with real RS settings and may face criticism, particularly Condition 6; we address some anticipated criticisms regarding this condition in Section 3.4. The Condition 2 is linked to the requirement that the simulation process must be fast enough so it can complete before any new item is introduced in the repository (i.e. if the simulation is fast enough, then both requirements are satisfied). It should be noted that this is a common assumption in the evaluation methods because the measurement process is expected to be fast enough so that the accumulation of uncommitted changes (i.e., changes that happen after the evaluation starts and, thus, are not accounted by the evaluation) does not compromise the assessment. With regard to Condition 4, an analysis of this premise was developed in Section 2.4.

3.3.1 Surprise as a finite resource

As assumed in the Scenario 3.3, at any given time, a recommender system has a finite number of items in its repository (Condition 1) and every user has been exposed to at least one item (Condition 3). On the basis of these premises we argue that surprise is a finite resource in the system, as illustrates the following scenario:

Scenario 3.4 *User u has been exposed to **all but one** item in the repository, namely i.*
Initial condition:

- *Let I represent the set of items in the repository of the system.*
- *Let $N_u = \{i\}$ represent the set of items unknown to the user u.*
- *Let $E_u = I \setminus N_u$ represent the set of items known to the user u.*

Interaction:

- *The system produces for the user u the recommendation list $L = (i)$.*
- *The user u consumes the item i (Condition 6 of the Scenario 3.3).*

Final condition:

- *The total amount of surprise the system produced to user u is $S_i(i, E_u)$.* ■

In this scenario, the amount of surprise available to the user u is finite[3] and corresponds to the surprise of the item i, because this is the only item still unknown to them. As a side note related to Condition 5, it must be noted that recommending a known item $k \in E_u$ would not produce any surprise according to the definition of surprise of an item (Equation 11) and the definition of a metric or distance (Definition 2.6.1), because:

$$S_i(k, E_u) = \min_{j \in E_u} dist(k, j) = dist(k, k) = 0 \leq dist(a, b), \forall a, b \in I.$$

It is possible to build on the previous scenario to allow for $|N_u| = 2$:

Scenario 3.5 *User u has been exposed to **all but two** items, namely i and j.*
Initial condition:

- *Let I represent the set of items in the repository of the system.*
- *Let $N_u = \{i, j\}$ represent the set of items unknown to the user u.*
- *Let $E_u = I \setminus N_u$ represent the set of items known to the user u.*

Interaction:

- *The system produces for the user u the recommendation list $L = (i, j)$.*
- *The user u consumes the item i and then the item j (Condition 6, Scenario 3.3).*

Final condition:

- *The total amount of surprise the system produced is $S_i(i, E_u) + S_i(j, E_u \cup \{i\})$.* ■

The term $E_u \cup \{i\}$ accounts for the fact that item i was known to user u when item j was presented to them. This is important because, as seen in Section 3.2, the order in which items are recommended may produce a different total amount of surprise. It is possible to build on the previous scenario to generalise how to compute the total amount of surprise that a sequence of items can offer to a user u in cases where $|N_u| \geq 1$:

Scenario 3.6 *User u has been exposed to **all but m** items, namely $i_1, i_2, \ldots, i_m$.*
Initial condition:

- *Let I represent the set of items in the repository of the system.*
- *Let $N_u = \{i_1, i_2, \ldots, i_m\}$ represent the set of items unknown to u.*

[3] It is finite because S_i is defined using a distance or metric, whose codomain is $\mathbb{R}_+$, and $+\infty \notin \mathbb{R}$.

- *Let $E_u = I \setminus N_u$ represent the set of items known to the user u.*

Interaction:

- *The system produces for the user u the recommendation list $L = (i_1, i_2, \ldots, i_m)$.*
- *The user u consumes each item in L in the order they rank in the list (Condition 6 of the Scenario 3.3).*

Final condition:

- *The total amount of surprise the system produced for the user u can be calculated by the following recursive function (surprise of a sequence):*

$$S_s(L, E_u) = \begin{cases} S_i(h, E_u) + S_s(t, E_u \cup \{h\}), & \text{if } |L| > 0, \\ 0, & \text{if } |L| = 0. \end{cases} \tag{35}$$

where h represents the head of the sequence L, namely L_1, and t represents its remaining items, $(L_2, \ldots, L_m)$. ■

Therefore, independently of the order in which the m items are recommended to the user u, the surprise of a sequence $S_s(seq, E_u)$ is always a sum of $m + 1$ finite values and, thus, is itself a finite value. As the number of users is also bounded, then the total amount of surprise a system can offer to its users is finite.

3.3.2 The potential surprise

As shown in the previous sections, the amount of surprise a system can produce for a user u is finite, and that amount depends on the order in which the items are presented to the user. These results inevitably lead to the question about the existence of a sequence of items that would maximise the amount of surprise to an arbitrary user. We argue that the maximum amount of surprise a system can offer to the user u must correspond to the surprise obtained by some sequence that contains all of the items unknown to the user u. In other words, this sequence is a permutation of the elements in N_u. In view of this, we define the maximum potential surprise as:

$$S_{pmax}(N_u, E_u) = \max_{tseq \in permut(N_u)} S_s(tseq, E_u), \tag{36}$$

where $permut(N_u)$ is the set of permutations of items in N_u, and thus $tseq$ represents any sequence that contains all of the items in N_u, with no repeating items; we refer to such a sequence as a **total sequence**.

The maximum potential surprise claims that there is at least one permutation of the items in N_u that maximise the surprise for the user u. This amount of surprise can be interpreted as the "stock of surprise" that the system can hold to the user u (i.e. the maximum amount of surprise the system can offer to an arbitrary user at any specific point in time). We qualified this surprise as being "potential" to make it clear that assessing the amount of surprise the system can offer to a user is, in essence, a counterfactual exercise: what would the resulting surprise be if the items that are unknown to the user at this moment were presented to them in a specific order[4].

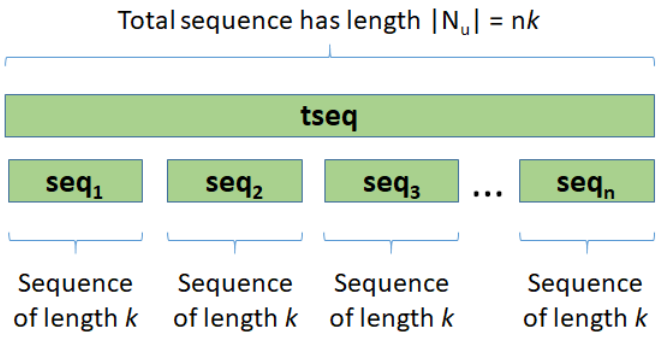
Analogously, we argue that the minimum surprise a system can offer to the user u must correspond to the surprise obtained by some total sequence (i.e., a permutation of all the items unknown to the user u), and we define the minimum potential surprise as:

$$S_{pmin}(N_u, E_u) = \min_{tseq \in permut(N_u)} S_s(tseq, E_u). \tag{37}$$

As seen, the concept of total sequence is basic to the definition of S_{pmax} and S_{pmin}. However, it may be argued that a real recommender system does not produce a recommendation list composed of all items unknown to a user. In response, we would point out that, on the basis of the Condition 6 of the Scenario 3.3, generating multiple lists of length $k < |N_u|$ to a user u has the same resulting surprise as generating a single list obtained by orderly concatenating all of the lists of length k, which corresponds to the definition of a total sequence. For example, suppose that a total sequence $tseq$ contains $nk = |N_u|$ items. In addition, as illustrated in Figure 8, suppose $tseq$ is orderly sliced into n sequences of length k: the first sequence, seq_1, contains the first k elements of $tseq$, the second sequence, seq_2, contains the following k elements of $tseq$, and so on. Then, exposing a user (proxy agent) to the sequences in the order they rank in $tseq$ produces the same effect as exposing them to $tseq$, because $tseq = seq_1, seq_2, \ldots, seq_n$, and the agent consumes the items in the order they rank in each sequence.

[4] The concept of potential surprise described here does not hold close resemblance to the concept of potential surprise in Economics, known as George Shackle's Potential Surprise Theory (PST).

Figure 8 – A total sequence and an order-preserving concatenation of its slices.



Source: Andre Paulino de Lima, 2019

Finally, note that the maximum potential surprise defines an upper bound to how much surprise a system can offer to a user, what answers the first research question (RQ1).

3.3.3 The normalised surprise of a total sequence

Once the maximum and the minimum amounts of surprise a system can offer to a user have been defined, we use these limits to create a scale to assess the surprise of a total sequence. We define the normalised surprise of a total sequence $tseq$ as:

$$S_{sn}(tseq, E_u) = \frac{S_s(tseq, E_u) \quad - S_{pmin}(F(tseq), E_u)}{S_{pmax}(F(tseq), E_u) - S_{pmin}(F(tseq), E_u)}, \tag{38}$$

where the predicate $F(seq)$ converts a sequence into a set comprised of the elements in the sequence, as required by S_{pmax} and S_{pmin}:

$$F(seq) = \{seq_j, \forall j \in 1 \ldots |seq|\}. \tag{39}$$

The normalised surprise of a total sequence produces a score within the interval $[0, 1]$. If $S_s(tseq, E_u) = S_{pmax}(F(tseq), E_u)$, then $S_{sn}(tseq, E_u) = 1$ and $tseq$ is a total sequence that maximises the surprise for the user u. On the other hand, if $S_s(tseq, E_u) = S_{pmin}(F(tseq), E_u)$, then $S_{sn}(tseq, E_u) = 0$ and $tseq$ offers the minimum amount of surprise to the user. It must be noted that the predicate S_{sn} is not defined when $S_{pmin}(F(tseq), E_u) - S_{pmax}(F(tseq), E_u) = 0$. In this situation, it holds that:

$$S_{pmin}(F(tseq), E_u) = S_{pmax}(F(tseq), E_u) \iff S_s(tseq, E_u) = v, \forall\, tseq \in permut(N_u),$$

which means that, in this scenario, all permutations of the items in N_u produce the same surprise. Moreover, it may be argued that v is the maximum and the minimum value

of potential surprise simultaneously. For practical reasons, we take account that $tseq$ is composed of new items, and convention that $S_{sn}(tseq, E_u) = 1$ in case of a singularity.

3.3.4 Computational costs and approximations

Real recommender systems have a huge number of items that are unknown to any given user. Since calculating the S_{pmax} requires evaluating the surprise of $|N_u|!$ total sequences, its exact computation is not tractable. However, an approximation to S_{pmax} (Equation 36) can be obtained by means of a greedy estimation strategy:

$$\hat{S}_{pmax}(N_u, E_u) = \begin{cases} S_i(i^*, E_u) + \hat{S}_{pmax}(N_u \backslash \{i^*\}, E_u \cup \{i^*\}), & \text{if } |N_u| > 0, \\ 0, & \text{if } |N_u| = 0, \end{cases} \tag{40}$$

with,

$$i^* = \arg\max_{i \in N_u} S_i(i, E_u). \tag{41}$$

As defined in Equation 41, i^* is the most surprising item in N_u with respect to E_u. Computing an approximation to S_{pmax} consists of successively finding the most surprising item and adding up its contribution to the total amount of surprise. The same technique can be used to obtain an approximation for S_{pmin} (Equation 37):

$$\hat{S}_{pmin}(N_u, E_u) = \begin{cases} S_i(i_*, E_u) + \hat{S}_{pmax}(N_u \backslash \{i_*\}, E_u \cup \{i_*\}), & \text{if } |N_u| > 0, \\ 0, & \text{if } |N_u| = 0, \end{cases} \tag{42}$$

with,

$$i_* = \arg\min_{i \in N_u} S_i(i, E_u). \tag{43}$$

As defined in Equation 43, i_* is the least surprising item in N_u with regard to E_u. We can now define an approximation to S_{sn} (Equation 38) using the approximations for the maximum and minimum potential surprises (Equations 40 and 42, respectively)[5]:

$$\hat{S}_{sn}(tseq, E_u) = \frac{S_s(tseq, E_u) \quad - \hat{S}_{pmin}(F(tseq), E_u)}{\hat{S}_{pmax}(F(tseq), E_u) - \hat{S}_{pmin}(F(tseq), E_u)}, \tag{44}$$

where the predicate $F(tseq)$ is described in Equation 39.

The adoption of a greedy approximation strategy has benefits and drawbacks. The main benefit is that it is simple, intuitive, and easy to implement. The main drawback

[5] The function $\hat{S}_{sn}$ is not defined when its denominator is zero. Following the decision taken for a similar situation with the function S_{sn}, in case of singularity we assume that $\hat{S}_{sn}(seq, E_u) = 1$ by convention.

is that it does not offer theoretical bounds to the quality of its approximations, so no guarantees can be assumed for the quality of the approximations obtained by $\hat{S}_{pmax}$ and $\hat{S}_{pmin}$. This risk is empirically addressed in Experiment 1 (Section 4.1), in which the quality of the approximations to the maximum and minimum surprises that the system can offer to a user is assessed.

3.3.5 The normalised surprise of a recommendation list

Up to this point, we have focused on estimating the total amount of surprise a recommender system can offer to an arbitrary user. The approach required finding a total sequence ($|tseq| = |N_u|$) that maximises the potential surprise for them. We now turn to the problem of estimating the maximum amount of surprise the system can embed in a sequence that does not necessarily contain all the items unknown to a user. Now suppose that a sequence seq contains k items, with $0 < k \leq |N_u|$. Then, it must be the case that:

- Any sequence must consist of an ordering of k arbitrary items in N_u, thus:

$$seq \in arrangements(N_u, k),$$

  where $arrangements(N_u, k)$ is the set of k-arrangements of items in N_u.
- The maximum amount of surprise that the RS can embed in a such a sequence is:

$$S'_{pmax}(N_u, E_u, k) = \max_{seq \in arrangements(N_u,k)} S_s(seq, E_u). \tag{45}$$

- The minimum amount of surprise that the RS can embed in such a sequence is:

$$S'_{pmin}(N_u, E_u, k) = \min_{seq \in arrangements(N_u,k)} S_s(seq, E_u). \tag{46}$$

- The normalised surprise of a sequence of length k can be defined as[6]:

$$S'_{sn}(seq, E_u, k) = \frac{S_s(seq, E_u) - S'_{pmin}(F(seq), E_u, k)}{S'_{pmax}(F(seq), E_u, k) - S'_{pmin}(F(seq), E_u, k)}. \tag{47}$$

It should be noted that the maximum potential surprise of a sequence of length k, $S'_{pmax}(N_u, E_u, k)$, addresses our second research question (RQ2), because it represents an upper bound to the surprise that a system can embed in a recommendation list of length k. However, since calculating S'_{pmax} or S'_{pmin} requires evaluating the surprise of

[6] Again, in case of singularity we assume that $S'_{sn}(seq, E_u, k) = 1$ by convention.

$\frac{|N_u|!}{(|N_u|-k)!}$ sequences, which is not feasible in realistic settings, we resort again to a greedy approximation strategy to obtain the estimates for these predicates:

$$\hat{S}'_{pmax}(N_u, E_u, k) = \begin{cases} S_i(i^*, E_u) + \hat{S}'_{pmax}(N_u \setminus \{i^*\}, E_u \cup \{i^*\}, k-1), & \text{if } k > 0, \\ 0, & \text{if } k = 0, \end{cases} \tag{48}$$

$$\hat{S}'_{pmin}(N_u, E_u, k) = \begin{cases} S_i(i_*, E_u) + \hat{S}'_{pmin}(N_u \setminus \{i_*\}, E_u \cup \{i_*\}, k-1), & \text{if } k > 0, \\ 0, & \text{if } k = 0. \end{cases} \tag{49}$$

In these equations, i^* is the most surprising item with respect to E_u, and i_* is the least surprising item, as defined in Equations 41 and 43, respectively.

We can now define an approximation to S'_{sn} (Equation 47) using the approximations for the maximum and minimum potential surprises for sequences of length k (Equations 48 and 49, respectively)[7]:

$$\hat{S}'_{sn}(seq, E_u, k) = \frac{S_s(seq, E_u) \quad - \hat{S}'_{pmin}(seq, E_u, k)}{\hat{S}'_{pmax}(seq, E_u, k) - \hat{S}'_{pmin}(seq, E_u, k)}. \tag{50}$$

As mentioned earlier, the main drawback of adopting a greedy strategy is that it does not offer theoretical bounds to the quality of its approximations, so no guarantees can be extended to the quality of the approximations obtained from $\hat{S}'_{pmax}$ and $\hat{S}'_{pmin}$. This risk is empirically addressed in Experiment 2 (Section 4.2), in which the quality of the approximations to the maximum and minimum surprises that the system can embed in a recommendation list is assessed.

3.4 Revisiting a premise of the model of potential surprise

The Condition 6 of Scenario 3.3 entails an assumption on how a real user reacts when presented to a recommendation list L: the user promptly consumes each item in the order it is ranked in the list. It may be argued that this is a questionable assumption because the expected user behaviour in real settings is that they will only consume a small fraction of the items that were recommended to them (NGUYEN et al., 2014). The following analysis aims to clarify the impact that the violation of this condition exerts on the assessment of surprise and on the internal validity of the theoretical model:

[7] Again, in case of singularity we assume that $\hat{S}'_{sn}(seq, E_u, k) = 1$ by convention.

- Preliminaries: let $userSurprise$ represent a quantitative measure of surprise obtained by a interviewing a user with respect to their experience of consuming a recommendation list L', and let $\hat{S}'_{sn}$ represent an estimate for $userSurprise$ obtained by the model of surprise underlying the $\hat{S}'_{sn}$ metric.
- A user may fail to notice or to follow a recommendation list L. In this case, the list of items consumed by the user u corresponds to an empty list L', and thus: $userSurprise(L', E_u) \approx \hat{S}'_{sn}(L', E_u, 0) = 0 \leq \hat{S}'_{sn}(L, E_u, |L|)$.
- A user may consume only a few items in L. In this case, the sequence of items consumed by the user corresponds to a list $L' \in arrangements(L, l)$, with $l \leq k$. Take $v = \hat{S}'_{sn}(L', E_u, |L'|) \approx userSurprise(L', E_u)$. We argue that if $l \ll k$, then the inequality $v \leq \hat{S}'_{sn}(L, E_u, |L|)$ it is expected to be true. However, the inequality may become false as l approaches k or as triangular asymmetries accumulate[8].
- A user may consume all items in L, but in a order that is different from the order in which the items are ranked in the list L. This situation is covered by the previous case, taking $l = k$.

In view of these results, we argue that the approximate normalised surprise of a sequence, $\hat{S}'_{sn}$, provides an upper bound to the real user surprise in most situations in which the Condition 6 is violated. Evidences that support this expectation can be found in Nguyen et al. (2014), which showed that about 30% of the users in the study never took recommendations from the system. In addition, the results also suggest that from the remaining users, 70% consumed on average just 1 item from each recommendation list containing 15 items, and 30% consumed on average 7.5 items per recommendation list[9]. If the $\hat{S}'_{sn}$ metric is an upper bound to the user surprise, then a change in system-level surprise estimated with this metric represents a true change in the capacity of the system to embed surprise in its recommendations, given that variations in user consumption habits along the time are isolated from the estimative.

[8] A distance function may not satisfy the triangle inequality (Axiom A4). In this case, the length of a path connecting three points (abc) may be smaller than the path connecting (ac).

[9] The study was conducted with 1,405 users from the movie recommendation system Movielens. The estimates presented here were obtained from averaged data reported on the study: a) each user has 12 rating blocks on average; b) each rating block contains 10 ratings, and cover roughly 3 months of the user rating history; c) on average, a user in the "Intermediate Group" ($N = 689$) has 3 rating blocks that contain at least one recommended item each; d) on average, a user in the "Following Group" ($N = 286$) has 9 rating blocks that contain at least one recommended item. To compute the estimates, we assumed that each user accesses the recommendations page ("Top Picks for You") every two weeks.

3.5 Adapted evaluation method

Since the $\hat{S}'_{sn}$ metric only estimates the surprise of a recommendation list, we need an evaluation method to organise the assessment of an RS with regard to this metric. We adopt the approach employed by Kaminskas and Bridge (2014) and adapt their evaluation method, described in Section 2.8.2, to employ the $\hat{S}'_{sn}$ metric. The fundamental ideas are retained, such as drawing a large sample of unknown items and allowing the user preference model to select those items that are more promising according to some application objective. Thus, it remains compatible with top-N evaluation tasks. In addition, it can be merged with other off-line methods based on the one-plus-random algorithm so that a single set of recommendation lists produced for the test users can be evaluated according to different metrics. This feature reduces the effect of randomness related to the creation of those lists in the final measurements and allow for more precise comparative studies. The changes that were introduced aim to make the evaluation method more flexible with regard to the partitioning of the dataset, and to the range of preference models that can be employed.

The adapted method is described in the Algorithm 4: it has five parameters, namely: a sample of users (`U`), the set of items in the repository (`I`), user ratings (`Ratings`), the length of a recommendation list (`topN`), and an external procedure (`metamodel`) that, given the ratings produced by the users, induces a model Θ that computes a score for an arbitrary, unknown item, as was seen in Sections 2.7 and 2.8.

In line 3, a user preference model is induced from the ratings, and possibly from content data. In line 5, E_u is assigned to the set of items known to the user u (the predicate I is the same used in the Algorithm 3), and in line 6, N_u is assigned to the set of items unknown to that user. In line 7, L_1 is assigned with a set containing 1,000 unknown items randomly drawn from N_u. In line 8, the list with candidate items L_1 is fed to the recommendation algorithms that, in turn, returns the recommendation list, which is assigned to variable *seq*. In line 9, the approximate normalised surprise of the recommendation list *seq* is computed and accumulated. Finally, in line 11, the algorithm returns the mean normalised surprise obtained from the sample of users `U`.

This evaluation method addresses our third research question (RQ3), since the method applies the $\hat{S}'_{sn}$ metric to an RS to obtain an estimate of how much of the available surprise has been embedded in recommendation lists produced by the system for its

Algorithm 4 Off-line evaluation method to estimate surprise of an RS using $\hat{S}'_{sn}$

```
1: procedure METHOD(U, I, Ratings, topN, metamodel)
2:     acc ← 0
3:     Θ ← metamodel( Ratings )
4:     ∀u ∈ U :
5:         E_u ← I(Ratings, {u})
6:         N_u ← I \ E_u
7:         L_1 ← random( N_u, 1000 )
8:         seq ← basic-recommendation-method(u, L_1, topN, Θ)
9:         acc ← acc + Ŝ'_sn(seq, E_u, topN)
10:    surprise ← acc / |U|
11:    return surprise
```

Source: Andre Paulino de Lima, 2019

users. The result obtained is in the $[0, 1]$ range, which is defined by the $\hat{S}'_{sn}$ metric. If the result is zero, then the system has embedded the minimum amount of surprise in each recommendation list that was produced for the users in the test sample `U`. If the result is one, then the system has embedded the maximum amount of surprise in each recommendation list that was produced during the evaluation. Assuming that a) the sample of users `U` is representative of the user population, and b) this past behaviour emerges from some stationary characteristics of the RS, then measurement obtained is a good estimate for the future behaviour of the system (i.e. the performance of the system in the future is expected to be similar to what has been observed in the measurement).

It must be noted that the computational cost of this proposed evaluation method is much higher than the cost of the method proposed by Kaminskas and Bridge (2014) owing to the cost of computing the $\hat{S}'_{sn}$ metric. The one plus random method for surprise has a computational cost that is proportional to $\texttt{topN} \times cost(S_i)$ (line 10 of the Algorithm 3), while the proposed method has a computational cost around $(2|L_1| + 1) \times \texttt{topN} \times cost(S_i)$ (line 9 of the Algorithm 4), because:

- Evaluating the surprise of the list, S_s, demands `topN` executions of $S_i(i, E_u)$.
- Evaluating the approximate maximum potential surprise of the list, $\hat{S}'_{pmax}$, demands about $\texttt{topN} \times |L_1|$ executions of $S_i(i, E_u)$ to perform `topN` greedy searches for the most surprising item i^* in the list of candidate items L_1.
- Evaluating the approximated minimum potential surprise of the list, $\hat{S}'_{pmin}$, also demands $\texttt{topN} \times |L_1|$ executions of $S_i(i, E_u)$.

The real computational cost of this method depends on the average size of E_u in a user population and, thus, may vary depending on the data distribution in the repository of the RS under evaluation. In contrast, since the proposed method does not need that the dataset be divided in different partitions, as the Algorithm 3 does, it tends to spend less computational resources in data selection tasks, but that also depends on the data distribution within the RS. An empirical, comparative analysis of execution time required by both evaluation methods will be presented in Section 4.4.

3.6 Closing remarks

In this chapter, we addressed the first three questions defined for this project:

RQ1 Are there limits to how much surprise a recommender can offer to a user? Yes, there are. This question was addressed by the development of the S_{pmax} metric, which is based on the potential surprise model described in Section 3.3. A greedy approximation to its exact value can be obtained by means of the $\hat{S}_{pmax}$ metric.

RQ2 Are there limits to how much surprise can be embedded in a recommendation? Yes, there are such limits. This question was tackled by the development of the S'_{pmax} metric, which computes the maximum amount of potential surprise that can be embedded in a recommendation list of arbitrary length. An approximation to its exact value using a greedy strategy can be computed by the $\hat{S}'_{pmax}$ metric.

RQ3 If these limits exist, is it possible to use them to create a measurement scale up in which the performance of a system can be measured? Yes; a scale was created using the minimum and maximum potential surprise of a recommendation list, and the metric S'_{sn} was defined in terms of these limits. An approximation to its exact value using a greedy strategy can be calculated using the $\hat{S}'_{sn}$ metric. It was argued that it provides an upper bound estimative of the surprise of a real user when exposed to a recommendation list, in despite of the expected violations of the Condition 6 of Scenario 3.3, on which the theoretical model is founded. Finally, to obtain a system-level estimation, an off-line evaluation method was proposed in Algorithm 4.

The metrics proposed in this section were presented in an informal language to promote a fluid progression of ideas, but a formal description is available in Appendix A.

4 Experiments

In this chapter, we adopt an empirical approach to address our fourth research question (RQ4), which inquires about the effect that choices on item representation and item comparison exert on surprise. However, before tackling this question, we need to confirm the soundness of one premise that was assumed in the theoretical model regarding our ability to estimate the potential surprise. In addition, we also compare the proposed evaluation method to the current state-of-the-art method in the literature. In view of this, the four experiments reported in this section are roughly organised in three veins:

- Assess the soundness of a premise: the exact value of the maximum or the minimum surprise of a recommendation list cannot be assessed in real-world settings, but an estimate of these values can be obtained by a greedy approximation, as seen in Section 3.3.4. Evaluating the quality of these approximations is the aim of the experiments reported on Sections 4.1 (Experiment 1) and 4.2.5 (Experiment 2), which respectively explore a synthetic and a grounded recommender system. The latter uses the Movielens 1M dataset as its repository, and this dataset is described in Section 4.2.1. The process employed to extract item vectors from the dataset is described in Section 4.2.2. In addition, the Experiment 2 also seeks to determine if the Movielens-1M dataset has any limitations that need to be amended to support its use in this project.

- Address RQ4: The experiment reported on Section 4.3 (Experiment 3) addresses the remaining research question. A factorial experiment design is adopted to assess the effect that item representation and item comparison models exert on the capacity of a recommender system to embed surprise in its recommendations.

- Compare proposed and current evaluation methods: results obtained by applying the proposed method and the current state-of-the-art method are compared in Section 4.4 (Experiment 4), which reports the difference in results and in execution times.

This chapter ends with a discussion of how the evidences from the experiments support the theoretical model of potential surprise developed in this work (Section 4.5). The code and data used in these experiments were made available at this github repository: https://github.com/andreplima/surprise-in-recsys.

It should be noted that the experiments are reported in a narrative pattern that is composed of five pieces: a) in "Method", an outline of the experiment is presented; b) in "Dataset", the raw data and the sampling method that were employed in the experiment are described; c) the "Procedure" presents a routine-like summary of the tasks that are carried out in the experiment, starting with a description of the input parameters, and ending with a description of the measurements that are collected; d) in "Variations", the set of input parameters employed to execute the Procedure are enumerated; and e) In "Results and discussion", the results obtained by executing the Procedure for each Variation are shown and analysed; there is an exception in Experiment 3, in which the results are shown in Section 4.3.2, the discussion of the results is presented in Section 4.3.3, and there is an additional discussion about the patterns found in the results in Section 4.3.4.

4.1 Experiment 1: evaluating approximations to the potential surprise of a system

As seen in Section 3.3.4, obtaining exact values for the maximum and minimum potential surprise is an intractable problem, but it is possible to obtain an estimate to these values by using a greedy approximation algorithm. The objective of this experiment was to assess the quality of this approximation by evaluating the difference between exact and approximated values for potential surprise of a system. Because of the exponential growth in the computational cost of obtaining exact solutions in repositories with more than just a few items, this experiment uses a synthetic (toy) dataset.

Method: compute exact values of the maximum and minimum potential surprises for a user; compute greedy approximations for the maximum and minimum potential surprises for the same user; compare differences between exact and approximate values.

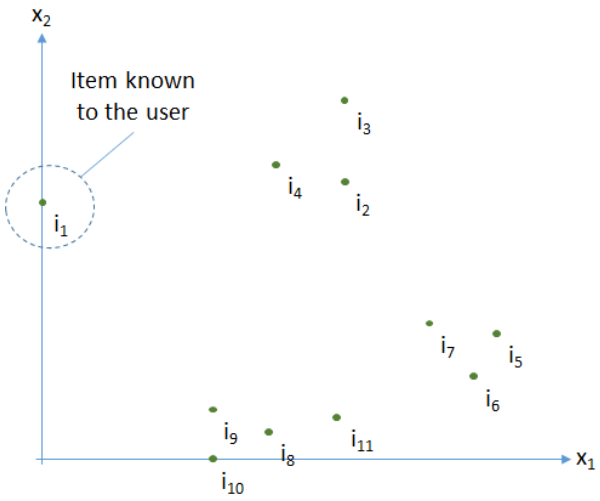
Dataset: the synthetic dataset represents the repository of an RS that has a single user, eleven items – labelled $\{i_1, \ldots, i_{11}\}$, and one rating (for item i_1). Items are represented as vectors in $\mathbb{R}^2$, for easier visualisation, and are arbitrarily distributed, as described in Figure 9. Despite of the arbitrary choices of number of items and their distribution, it must be noted that: a) the item vectors are compatible with count-based representation models (Models C and U in Section 2.5), since they only contain non-negative elements, b) as item

vectors are not normalised, effects of normalisation can manifest, c) as some item vectors contain zero-valued elements, effects of smoothing can manifest, and d) the number of items was chosen so recommendation lists of length 10 could be produced, which is a size of recommendation lists typically explored in experiments in the literature[1]. Thus, as the set of items unknown to the user is $N_u = \{i_2, \ldots, i_{11}\}$, there can be $|N_u|! = 10!$ sequences.

Procedure: given a distance function, the surprise of each possible sequence composed of the items that are unknown to the user is computed according to the definition of surprise of a sequence, $S_s(seq, E_u)$, in Equation 35. The maximum and minimum values obtained correspond to the exact values of the maximum and minimum potential surprise, respectively defined in Equations 36 and 37. These values are recorded. The greedy approximations to the maximum and minimum potential surprise are computed according to their definitions in Equations 40 and 42. These values are recorded and later, during the analysis of the results, are compared to their exact counterparts.

Figure 9 – Description of the item vectors in the fictitious RS.

Item	x_1	x_2
i_1	**0**	7.128
i_2	6.677	7.719
i_3	6.630	**10**
i_4	5.135	8.147
i_5	**10**	3.523
i_6	9.460	2.240
i_7	8.506	3.768
i_8	4.976	0.652
i_9	3.704	1.385
i_{10}	3.720	**0**
i_{11}	6.439	1.120



Source: Andre Paulino de Lima, 2019

Variations: as the surprise of a sequence, $S_s(seq, E_u)$, depends on a distance function, the Procedure was repeated using distance functions of diverse intuitions, as described in Section 2.6: the non-normalised Euclidean distance and the cosine distance (geometric

[1] Initial works on top-N recommendation tasks explored a range of values for N, such as $N \in [1, 20]$ (CREMONESI; KOREN; TURRIN, 2010). However, to promote commensurability of results from works relating surprise and recall, we adopted $N = 10$ (KAMINSKAS; BRIDGE, 2014).

intuition), the Jaccard distance (combinatorial), the Kullback-Leibler and Jensen-Shannon divergences (informational), and the Aitchison distance (compositional).

Results and discussion: the results are summarised in Table 3. In the table, values in the D_{max} and D_{min} columns correspond to the relative difference between exact and approximate values of maximum and minimum potential surprise, defined as:

$$D_{max} = 100 \times \frac{S_{pmax} - \hat{S}_{pmax}}{S_{pmax}},\ D_{min} = 100 \times \frac{\hat{S}_{pmin} - S_{pmin}}{S_{pmin}}.$$

Except for the cases of $\hat{S}_{pmax}$ in which the Euclidean distance or the Aitchison distance was used (these are the only unbounded distances), no difference between the exact and approximate computations was obtained[2]. The deviations occurred because $\hat{S}_{pmax}$ underestimated S_{pmax}. For the Euclidean distance, the difference was about 2%, and for the Aitchison distance the difference was about 0.5%.

Table 3 – Greedy approximations to potential surprise in an RS with a synthetic repository. Values of D_{max} or D_{min} represent the relative difference between the exact and the approximate potential surprise. Values under 0.01% are written as a dash.

Distance	Intuition	S_{pmax}	$\hat{S}_{pmax}$	$D_{max}(\%)$	S_{pmin}	$\hat{S}_{pmin}$	$D_{min}(\%)$
Euclidean	geometric	**37.684**	**36.948**	1.95	23.935	23.935	—
cosine	geometric	0.684	0.684	—	0.138	0.138	—
Jaccard	combinatorial	3.784	3.784	—	2.552	2.552	—
Kullback-Leibler	informational	2.449	2.449	—	0.437	0.437	—
Jensen-Shannon	informational	1.124	1.124	—	0.239	0.239	—
Aitchison	compositional	**7.133**	**7.097**	0.50	3.435	3.435	—

Source: Andre Paulino de Lima, 2019

Although these results do not support the general claim that a greedy algorithm will always achieve approximations as good as those obtained, they provide evidence that a local approximation may be a feasible approach to computing potential surprise in real settings. As this is a necessary condition to assessing potential surprise, it will be revisited in the Experiment 2, in which this experiment is performed using a realistic dataset.

4.2 Experiment 2: exploring the Movielens dataset

The main purpose of this experiment was to explore the Movielens dataset and identify possible limitations to its use in this project (e.g. missing attributes, entity without

[2] Though values are reported with three decimal digits of precision, we considered that results did not differ if they were indiscernible under standard IEEE 754 64-bit floating-point comparison.

required relationship). This section is divided into three parts: the first part describes the Movielens dataset and justifies the need to extend it for this project (Section 4.2.1); the second part presents the process that was used to extract item vectors and compute distance matrices from the dataset (Section 4.2.2); and the final part is an extension of the first experiment, but now samples of the Movielens dataset are used to assess the quality of greedy approximations to S'_{pmin} and S'_{pmax} (Section 4.2.5).

4.2.1 Extending the Movielens dataset

The Movielens-1M dataset (HARPER; KONSTAN, 2015) was created by a research group affiliated to the University of Minnesota (US) and contains data sampled from the online movie recommendation system Movielens[3], which is maintained by the same group. This dataset consists of ratings that were given by users who became active in the system in the year 2000 and contains approximately 1 million ratings attributed by 6,040 anonymised users to 3,883 movies.

The dataset comprised of three files, containing user data (USERS.DAT), movie data (MOVIES.DAT) and rating data (RATINGS.DAT). These files maintain a relational model whose attributes are described in Table 4. The users datafile contains demographic data, such as age, occupation, region of residence, and binary gender. The movie datafile records the title of each movie and a set of genres in which the movie fits. Finally, the ratings file stores tuples composed of a user, a movie, the rating given by the user to the movie, and a temporal reference (timestamp) to when the rating was created. The ratings collected in the dataset were created between April 2000 and February 2003.

As mentioned earlier, different categories of RS employ different data to promote personalisation: a CFRS exclusively employs ratings given by the users, but a CBRS uses the content combined with an item in addition to the ratings given by a specific user. To investigate how the theoretical model of potential surprise performs in both approaches, the Movielens 1M dataset was extended to combine a short textual description to each movie. This extension enables the adoption of representation models that can be applied to extract item vectors from item content, as was described in the Section 2.5. These textual descriptions were collected from the online movie recommendation system Movielens and

[3] Available at https://movielens.org.

Table 4 – Attributes of the extended MovieLens-1M dataset. The dataset contains 6,040 users, 3,883 items, 1,000,209 ratings and was extended with 3,804 short textual descriptions of the items. Each user has at least 20 ratings.

File	Attribute	Description
users	userID	User unique identifier.
users	gender	User's binary genger (M ou F).
users	age	User's age group. Domain: 1 (less than 18 yrs.), 18 (between 18 and 24 yrs.), 25 (between 25 and 34 yrs.), 35 (between 35 and 44 yrs.), 45 (between 45 and 49 yrs.), 50 (between 50 and 55 yrs.) e 56 (more than 55 yrs).
users	occupation	User's occupation. Domain varies between 0 and 20.
users	zip-code	Postal code of the user's residence, according to US Postal Service.
movies	movieID	Movie unique identifier.
movies	title	Movie title, which corresponds to the same title published in the online movies catalog IMDb.
movies	genres	The list of genres in which the movie fits, composed of any of the following 18 genres: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.
ratings	userID	Identifier of the user who submitted the rating.
ratings	movieID	Identifier of the rated movie.
ratings	rating	Rating given by the user to the movie. Domain: 1 to 5 stars.
ratings	timestamp	Rating creation date, represented as a positive integer number that indicates the number of seconds elapsed since January 1st, 1970, 00:00:00 (UTC).
moviedesc	movieID	Identifier of the movie.
moviedesc	description	Short description of the movie, obtained from the online movie recommender Movielens.

Source: Andre Paulino de Lima, 2019

were stored in the MOVIEDESC.DAT file, which is described in Table 4. The descriptions are in English and are similar to the blurbs usually found in the back of DVD or BD cases, which are written to increase the interest in the movie and do not contain spoilers. Therefore, these descriptions only provide a superficial narrative account of the movie and are not intended to expose the movie plot, as is typically the case of synopses.

4.2.2 Preprocessing the extended Movielens dataset

As was seen in Sections 2.5 and 2.6, the choice of representation model determines how item vectors are extracted from the RS repository, and the choice of distance function specifies how the dissimilarity between two item vectors is assessed. By adopting an

empirical approach to assess the effect that these choices exert on the evaluation of surprise, as required by our fourth research question, a process to apply different representation models and distance functions to the extended Movielens dataset is necessary. The main challenge of designing such a process is to identify the requirements that are simultaneously imposed by a representation model and a distance function: some requirements are common to all pairs of a representation model and a distance function, some requirements are specific to the representation model and others are particular to the distance function.

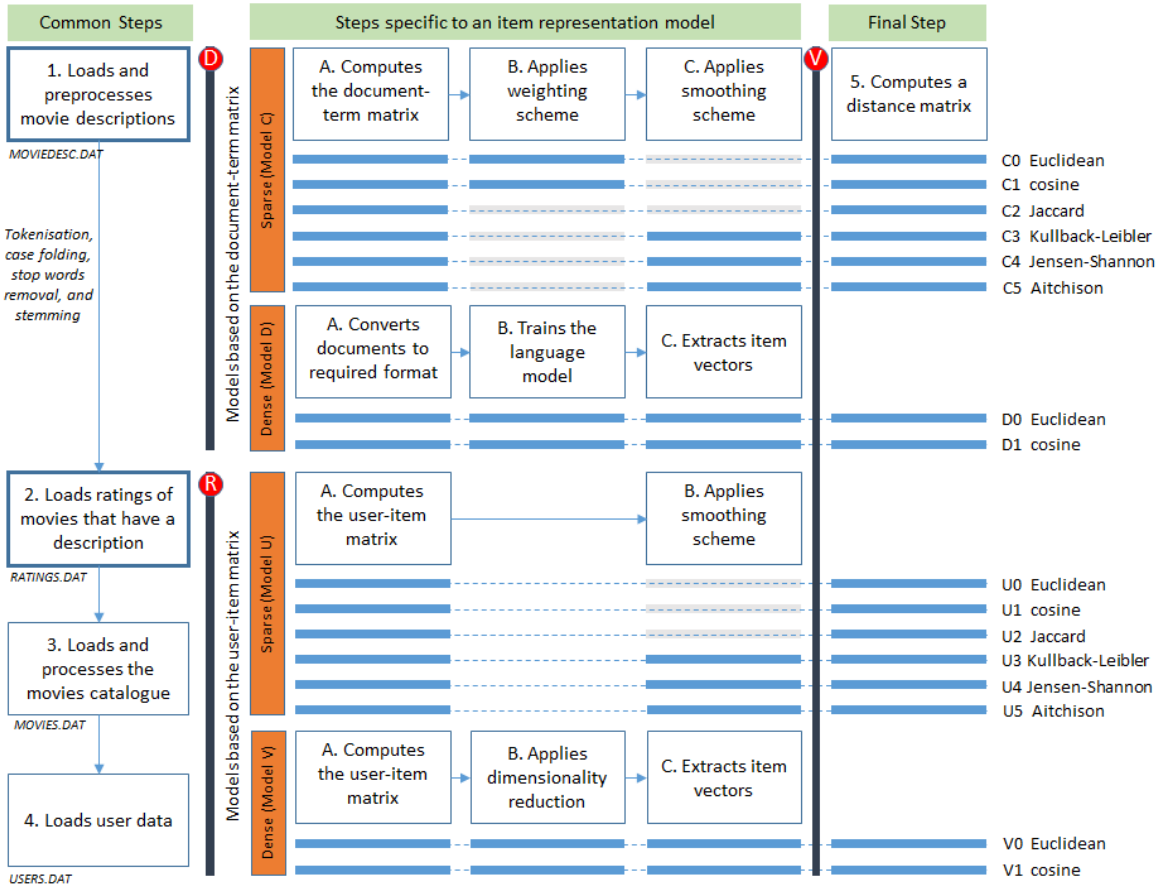
The process employed in this work is illustrated in Figure 10. The sequence of steps at the left of the diagram, under the label "Common Steps", cover the requirements that are common to all representation models, whereas the multiple sequences of steps at the middle, under the label "Steps specific to a representation model", cover the requirements that are particular to each item representation model. There is also a "Final Step", at the right of the diagram, which is also common to all representation models. The four item representation models described in Section 2.5 are explored, as well as six of the distance functions described in Section 2.6.

The Common Steps, which may have small variations depending on the representation model adopted, can be described as follows:

- Step 1 - Loads and preprocesses movies descriptions: the short textual descriptions that are combined with each item are submitted to tokenisation, case folding, stop words removal, and stemming. The count-based DSM (Model C) and the predictive DSM (Model D) exclusively use data that are created in this step: this dependence is represented by the vertical bar that separates Step 1 from the two horizontal sequences of steps that correspond to the processing required by these representation models (under the "Steps specific to an item representation model" label).

- Step 2 - Loads ratings of movies that have a description: the ratings given to the movies that have a valid textual description available are loaded. Both the sparse and dense user-item models (Models U and V, respectively) exclusively use data produced by this step and, again, this dependence is represented by the vertical bar that separates Step 2 from the two sequences of steps specific for these representation models.

Figure 10 – A process to extract item vectors from the Movielens dataset.



Source: Andre Paulino de Lima, 2019

- Step 3 - Loads and processes the movies catalogue: the attributes of an item are loaded. The title of a movie and its genre labels are loaded for convenience, but are not used in tandem with the short textual description recovered in Step 1.

- Step 4 - Loads user data: data from users that have rated at least one accepted item are loaded for convenience, but are not used in the remaining of the process.

This sequence of steps reinforces two restrictions: a) in Step 1, an item is rejected if it does not have a textual description, or if its textual description is too short or is not written in English, and b) in Step 2, an item is rejected if it does not have any ratings. If an item is rejected in any of these steps, then any attributes that may have been loaded are discarded. Therefore, rejected items will not be considered downstream in the process.

These restrictions aim at promoting equivalence across the experimental conditions under which the evaluation method will be applied. For example, items without a textual description can be properly handled by the representation models based on the user-item

matrix (Models U and V), but they pose a threat to the models based on the document-term matrix (Models C and D): if accepted, such items would be represented as zero vectors and, as a result, the similarity between any two of them would be the highest possible, which probably is not the case[4]. In the same vein, descriptions that are too short may not contain information to support adequate similarity assessment, and descriptions written in other languages would require translation into English to allow similarity assessment. Analogously, items with no ratings can be properly handled by the distributional semantic models (Models C and D), but they pose a threat to the user-item models (Models U and V): such items would be represented as zero vectors, bringing about the same problem that the items with no description raise in distributional semantic models.

Equivalence across experimental conditions plays an important role in the control of variability. In an experimental setting, the variation in the results can be classified into systematic or unsystematic variation. The systematic variation results from the manipulation of the controlled factors (e.g., choice of representation model and the distance function), whereas the unsystematic variation is due to uncontrolled factors, which includes factors that are unknown to the experimenter. Thus, reducing the unsystematic variation is a common objective in experiment design, and typically requires that all things must be equal across all experimental conditions (e.g. input data, characteristics of the environment in which the experiment is performed), except for the factors under analysis. In such a setting, it is reasonable to presume that the variation in the results that should be attributed to the input data is null, because the latter is the same across all conditions.

After the Common Steps have been applied, the resulting data corresponds to a mapping D, where D_i corresponds to a sequence of tokens extracted from document d_i, and a mapping R, where r_{ui} corresponds to the rating given by the user u to the item i. Item vectors are extracted from mappings D and R by the item representation models, which apply different sequences of steps in the process:

- Sparse document-term model or count-based DSM (Model C): it applies three steps, which were described in Section 2.5.1[5], and are illustrated in Figure 10. A document-term matrix $\mathbf{D}$ is computed in step A (using data from the mapping D), a weighting scheme is applied to the document-term matrix $\mathbf{D}$ in step B (e.g., tf-idf), and a

[4] Actually, zero vectors cannot be properly handled by some distance functions (e.g Aitchison distance).

[5] The Step A in Figure 10 corresponds to Steps 1 to 3 from Section 2.5.1, and Steps B and C in Figure 10 are detailed in Step 4 from that same section.

smoothing scheme is applied to it in step C (e.g., BMT with Perks prior). The horizontal bars under each step indicate that the distance function in that row requires the step to be executed. For example, the cosine distance requires steps A and B[6], the Jaccard distance requires only step A, and the Aitchison distance requires steps A and C.

- Dense document-term model or Predictive DSM (Model D): it has three steps, which were also described in Section 2.5.1. In step A, the mapping D is converted into the representation required by the language model (we adopted the `Doc2Vec` language model) (MIKOLOV et al., 2013b; ŘEHŮŘEK; SOJKA, 2010), then the language model is optimised to fit the data in step B, and the item vectors are extracted from the language model in step C. It must be noted that the only distance functions that are compatible with this representation model are the Euclidean and the cosine distances, as illustrated in Figure 10. As mentioned before, this particular Predictive DSM can be cast as a factorised model based on the document-term matrix (LEVY; GOLDBERG, 2014), and is classified here as a dense model.

- Sparse user-item model (Model U): it has two steps, which were described in Section 2.5.2. A user-item matrix $\mathbf{R}$ is computed in step A using data in the mapping R, and a smoothing scheme is applied to $\mathbf{R}$ in step B. Since this model produces compositional data, just as the count-based DSM, the same set of distance functions can be applied to the extracted item vectors.

- Dense user-item model (Model V): it has three steps, which were also described in Section 2.5.2. In step A, the user-item matrix $\mathbf{R}$ is computed, dimensionality reduction is applied to $\mathbf{R}$ using the PureSVD algorithm in Step B, and item vectors are extracted from the matrix $\mathbf{Q}$ produced by the algorithm in Step C.

After the "Specific Steps" have been executed, the resulting data corresponds to a matrix $\mathbf{V}$ in which each row $\mathbf{V}_i$ is a vector that encodes the item i according to the representation model adopted. The final step in the process, "5 - Computes a distance

[6] In Information Retrieval, the cosine distance is usually applied to tf-idf weighted vectors. Although this is not a requirement, since the cosine distance is well defined on unweighted item vectors, it is generally accepted that weighting increases its discriminative power. In this work, we followed this tradition and apply tf-idf weighting on item vectors from distributional semantic models, and extended this decision to the Euclidean distance, since this distance is also based on a geometric intuition.

matrix", is also common to all representation models and distance functions. It computes a distance matrix $\mathbf{M}$ such that $m_{ij} = dist(\mathbf{V}_i, \mathbf{V}_j)$, where $dist$ can be any distance function compatible with the representation model. The creation of a distance matrix has two benefits: first, it may improve the performance of the evaluation method, because the distance between each pair of items is computed only once; second, it elides irrelevant details from each representation model (e.g. vector dimensionality, choice of data structure), which allows the evaluation method to employ a single, generalised abstraction of distance.

As a final note, the codes in front of the distance functions in Figure 10 is a short abbreviation of the configuration that produced a particular distance matrix: the first character represents the item representation model (C for count-based DSM, D for predictive DSM, U for sparse user-item, and V for dense user-item model) and the second character represents the distance function employed (0 for the Euclidean distance, 1 for the cosine distance, 2 for the Jaccard distance, 3 for the Kullback-Leibler divergence, 4 for the Jensen-Shannon divergence, and 5 for the Aitchison distance). This code will later be employed to specify the configuration of an experiment.

4.2.3 Analysing results from the Common Steps

As mentioned earlier, the Movielens-1M dataset contains 3,883 items. For 3,804 of those, a short textual description was collected from the online movie recommender Movielens in September 2017. However, it was noted that some descriptions were too short, containing textual content such as "No overview found", "German Comedy" or "Directed by Goetz Grossmann". This problem motivated the restriction in Step 1: a textual description with less than 5 tokens after preprocessing must be rejected. This threshold was specified by visually inspecting the descriptions along with their respective list of tokens. Some exemplars of accepted and rejected items are presented in Table 5. These exemplars were selected to represent typical instances of different segments of the token count distribution[7]: the first segment represents the items that were rejected because they had no description available; the second segment represents the items that were rejected because the description was too short; the third segment represents accepted items with a token count below the first quartile ($Q_1 = 18$), the fourth segment represents the accepted

[7] In this sense, a typical instance of a segment is an item that has the median number of tokens per document within that segment.

items with token count between (and including) the Q_1 and Q_3 quartiles ($Q_3 = 40$), and the last segment represents the accepted items with token count above Q_3. In total, 3,643 items and 997,136 ratings were accepted, which corresponds to the rejection of 240 items (6.2% of total) and 3,073 ratings (0.3% of total). Since all explored configurations of a representation model and a distance function use the same set of items and set of ratings, the aimed equivalence of experimental conditions between configurations was achieved.

Table 5 – Examplars within each segment of the token per document distribution.

Segment	Item	Description
0 tokens	33	Wings of Courage (1995).
1 to 4 tokens	739	Honeymoon (1996). *German Comedy.*
5 to 17 tokens	364	The Lion King (1994). *A young lion cub named Simba can't wait to be king. But his uncle craves the title for himself and will stop at nothing to get it.*
18 to 40 tokens	589	Terminator 2: Judgment Day (1991). *Nearly 10 years have passed since Sarah Connor was targeted for termination by a cyborg from the future. Now her son, John, the future leader of the resistance, is the target for a newer, more deadly terminator. Once again, the resistance has managed to send a protector back to attempt to save John and his mother Sarah.*
41 or more tokens	1343	Cape Fear (1991). *Sam Bowden is a small-town corporate attorney. Max Cady is a tattooed, cigar-smoking, bible-quoting, psychotic rapist. What do they have in common? Fourteen years ago, Sam was a public defender assigned to Max Cady's rape trial, and he made a serious error: he hid a document from his illiterate client that could have gotten him acquitted. Now, the cagey, bibliophile Cady has been released, and he intends to teach Sam Bowden and his family a thing or two about loss.*

Source: Andre Paulino de Lima, 2019

4.2.4 Analysing results from the Final Step

After executing the Final Step for all compatible configurations, 16 distance matrices were produced. Figure 11 presents the histograms of pairwise distances obtained from these distance matrices. In the figure, each histogram represents the distribution of distances between pairs of item vectors obtained from a specific distance matrix. For example, the histogram at the upper left position presents the pairwise distance distribution for the config C0 (count-based DSM, Euclidean distance). For each point in the curve, its ordinate shows the number of entries of the distance matrix that stores the distance

indicated in the abscissa[8]. The results show that: a) some distributions are negatively skewed (skewness < -0.5): distributions from the Kullback-Leibler and Jensen-Shannon divergences, as well as those from Jaccard and cosine distances (except for config D1); b) some distributions are positively skewed (skewness > 0.5): distributions from the Euclidean distance (except for config D0); c) some distributions are nearly symmetric ($|\text{skewness}| < 0.5$): distributions from the Aitchison distance, plus configs D0 (Euclidean) and D1 (cosine)[9]. In addition, there is a positive correlation (+0.73, Pearson) between the skewness and the coefficient of variation in the distributions that were analysed, which means that the coefficient is expected to grow in so far as a distribution moves from being negatively skewed to being positively skewed. Finally, the embedding spaces adopted for dense representation models D and V have 100 dimensions[10].

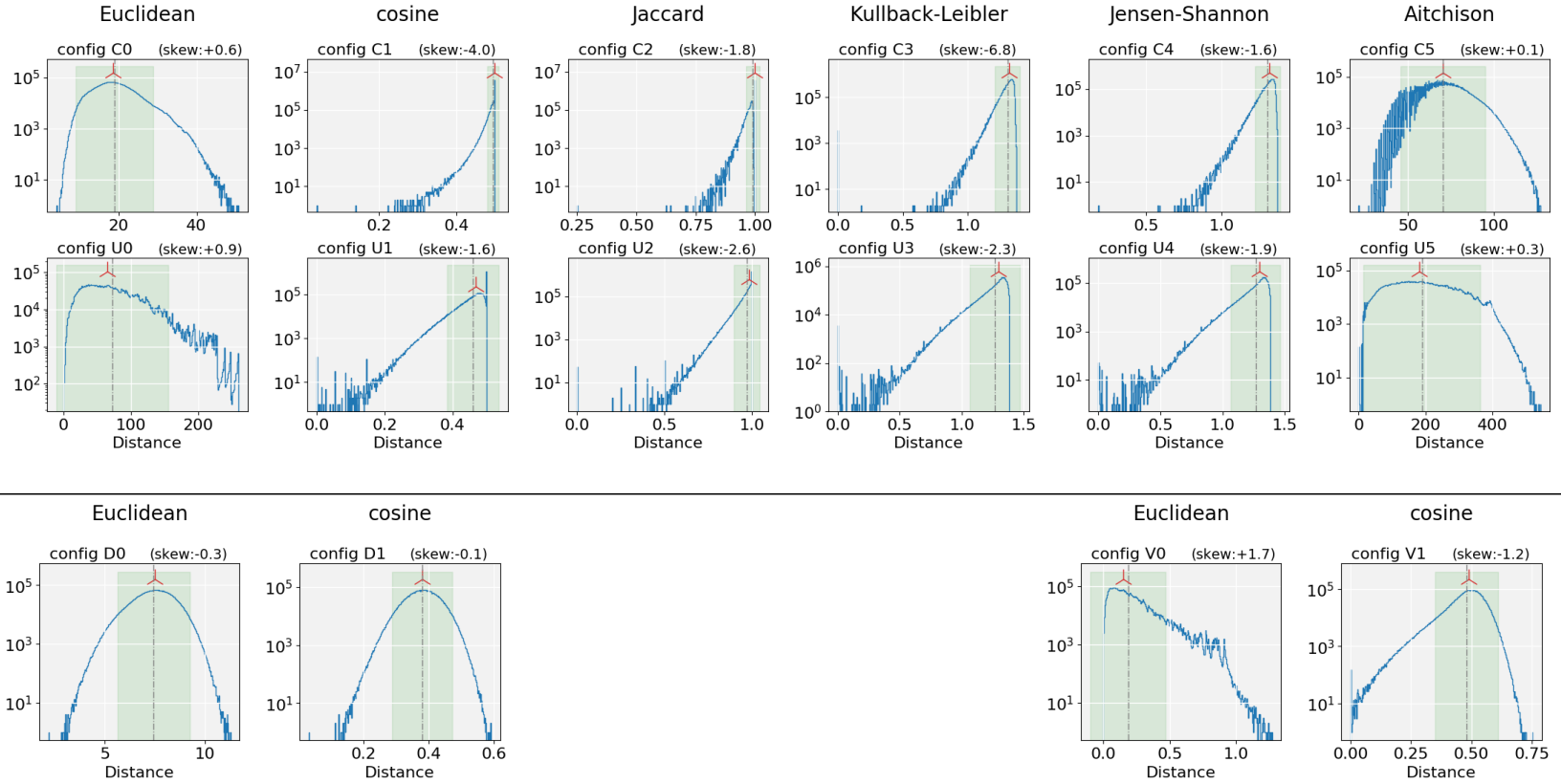
We use histograms of pairwise distances because they may provide qualitative insights about how a distance function discriminate items vectors. For example, suppose that we obtained a pairwise (Euclidean) distance distribution that is concentrated on a single distance value v that is different from zero: $dist(i,i) = 0, dist(i,j) = v, \forall i,j \in I, i \neq j$ (with finite I). This means that any two items i and j keep a distance v between them, and, thus, all item vectors are mutually equidistant. Such a distance function is detrimental to recommender systems using neighbourhood-based algorithms, because the task of finding the k-nearest neighbours of an item i, with $k < |I \setminus \{i\}|$, will only achieve multiple, ambiguous solutions. This distance function is also detrimental to the S_{sn} metric, because it becomes a singularity when all item vectors are mutually equidistant. Now, exploring the opposite extreme, suppose that we obtained a unit flat (Euclidean) distance distribution: $dist(i,i) = 0, dist(i,j) \neq dist(i,k), \forall i,j,k \in I, i \neq j \neq k$. In this case, the task of finding the k-nearest neighbours of an item i has a single solution for any $k \leq |I \setminus \{i\}|$, and the S_{sn} metric is always well defined. The distance functions explored in this work (and, in general, those of practical use), are in between these two extremes, and their distributions will be explored in the next experiment to reveal some intuitions about the relationship between system-level surprise and the skewness of its distance distribution.

[8] For symmetric distances, the item pairs (i,j) and (j,i) are equivalent and, because of this, they are counted just once in the histogram. The Kullback-Leibler divergence is the only exception to this.

[9] The distributions from configs D0 and D1 were nearly symmetric by design, and aimed at gathering a larger set of distribution profiles to be used in the next experiment (negatively and positively skewed, plus nearly symmetric distributions). It was achieved by manipulating the number of training epochs.

[10] The number of dimensions was chosen so our results can be compared with other works on surprise.

Figure 11 – Histograms of pairwise distances. Each histogram shows the pairwise distance distribution obtained for the config indicated in the upper left corner. The histograms obtained from Model C are in the first row, from Model U in the second row, and from Models D and V in the third row, left and right, respectively. Histograms that are in the same column are related to the distance function indicated in the column title. Within each histogram, the dashed vertical line represents the mean of the distribution, the shaded area around it covers two standard deviations ($\mu \pm 2\sigma$) and is proportional to the coefficient of variation ($cv = \sigma/\mu$), and the triangular marker ($\curlywedge$) represents the median. Distribution skewness is indicated in the upper right corner of each histogram. Note that the ordinate is in log scale.



Source: Andre Paulino de Lima, 2019

4.2.5 Evaluating approximations to the potential surprise of a sequence

Now that a real-world dataset is available, we resume the question approached in Experiment 1 and use this data to evaluate the quality of the approximations to S'_{pmax} and S'_{pmin} obtained by a greedy approach. Since it is not feasible to employ the whole dataset, small samples were collected.

Method: compute exact values of the maximum and minimum potential surprises of a sequence (recommendation list) drawn from samples of the Movielens dataset; compute the greedy approximations to maximum and minimum potential surprises of the same sequences; compute differences between exact and approximate values.

Dataset: samples were randomly drawn from the Movielens dataset. The sampling method consists of the following steps: a) a user is randomly selected and their rating history is sorted chronologically, b) the latest N items from the sorted history are taken as items for recommendation (N_u) and the remaining items are allocated as the user past experience (E_u), and c) the sample is defined as the (N_u, E_u) tuple. We adopted $N = 10$.

Procedure: given a config (item representation and distance function), a sample (N_u, E_u), and a positive integer (k) representing the target sequence length, all possible sequences of length k are generated from the sample (specifically from N_u). For each such sequence seq, its surprise value is obtained by evaluating $S_s(seq, E_u)$ according to the item representation and distance function specified in the config. The maximum and minimum surprise values obtained from these sequences are recorded. Greedy approximations to S'_{pmax} and S'_{pmin} are computed for the sample (N_u, E_u) and results are recorded.

Variations: the Procedure was applied to the combinations of 30 samples, 16 configs, and sequence lengths $k \in [2, N]$, resulting in 4,320 variations.

Results and discussion: the results obtained are summarised in Table 6. Values in the "max" and "min" columns correspond to the relative differences between exact and

approximate values of maximum and minimum potential surprise. For a specific sequence length k and the set of samples Ω, the relative differences are calculated as:

$$D_{max}(\Omega, k) = \frac{100}{|\Omega|} \sum_{(N_u, E_u) \in \Omega} \frac{S'_{pmax}(N_u, E_u, k) - \hat{S}'_{pmax}(N_u, E_u, k)}{S'_{pmax}(N_u, E_u, k)}, \text{ and}$$

$$D_{min}(\Omega, k) = \frac{100}{|\Omega|} \sum_{(N_u, E_u) \in \Omega} \frac{\hat{S}'_{pmin}(N_u, E_u, k) - S'_{pmin}(N_u, E_u, k)}{S'_{pmin}(N_u, E_u, k)}.$$

Thus, each value in a "*max*" or "*min*" column is calculated as the average of the relative differences obtained from 30 sequences of length k and 30 sequences of length $k+1$. For example, the value 0.34 in the second row (config C1) and fourth column ("*min*" for $n = 2; 3$) is the average relative difference obtained from 30 sequences of length 2 and 30 sequences of length 3. The results were grouped in this way with the sole purpose of reducing the number of entries in the table. In summary:

- The relative differences between exact and approximate values of potential surprise of a sequence were small: the largest relative difference obtained was 1.42% (for S'_{pmin}, config V1), which is close to 1.95%, the largest deviation obtained in Experiment 1.

- In the Averages row, there is a consistent pattern in that the relative difference for S'_{pmin} decreases as the sequence length increases. This pattern is also seen in the individual variations, with exceptions in the configs U1 and U3. However, a regression analysis of the raw results[11] indicates that a linear relationship among the representation model, the distance function, and the sequence length does not account for much of the variation in D_{min} ($R^2 < 0.05$) nor in D_{max} ($R^2 < 0.04$). The results of the regression analysis are reported in Table 7.

4.2.6 Summary of the results

All instances of each entity in the Movielens-1M dataset (user, movie, and rating) had the expected attributes and, thus, no strategy to fill in missing attributes was required. All instances observed the restrictions imposed by the relational data model and, thus, no duplicated instances or missing mandatory relationships were found. To allow the use of the dataset in a CBRS approach, short textual descriptions were collected for and

[11] The raw results is composed of tuples in the format: (*repmodel, distfn, length*, D_{min}, D_{max}).

Table 6 – Greedy approximations to potential surprise in a sample of the Movielens-1M. Values in the *max* or *min* columns are the relative difference between exact and approximate values of potential surprise. Dashes represent values under 0.01%.

Cfg	*n*	*max*	*min*	*n*	*max*	*min*	*n*	*max*	*min*	*n*	*max*	*min*	*n*	*max*	*min*
C0	2; 3	—	—	4; 5	—	—	6; 7	—	—	8; 9	—	—	10	—	—
C1	2; 3	—	0.34	4; 5	—	0.24	6; 7	0.01	0.08	8; 9	0.01	0.03	10	—	—
C2	2; 3	0.01	0.23	4; 5	0.03	0.13	6; 7	0.03	0.03	8; 9	0.02	—	10	0.02	—
C3	2; 3	—	0.17	4; 5	—	0.16	6; 7	0.01	0.11	8; 9	0.01	0.01	10	0.01	—
C4	2; 3	—	0.15	4; 5	0.02	0.13	6; 7	0.01	0.06	8; 9	—	0.04	10	—	—
C5	2; 3	—	—	4; 5	—	—	6; 7	—	—	8; 9	—	—	10	—	—
U0	2; 3	—	0.14	4; 5	—	—	6; 7	—	—	8; 9	—	—	10	—	—
U1	2; 3	—	0.71	4; 5	—	0.46	6; 7	—	0.54	8; 9	0.01	0.14	10	0.02	—
U2	2; 3	—	0.19	4; 5	—	0.13	6; 7	—	0.06	8; 9	—	—	10	—	—
U3	2; 3	—	0.51	4; 5	0.05	0.71	6; 7	0.03	0.61	8; 9	0.05	0.06	10	0.04	—
U4	2; 3	—	0.74	4; 5	0.01	0.54	6; 7	0.05	0.25	8; 9	0.05	0.06	10	0.05	—
U5	2; 3	—	0.33	4; 5	—	0.04	6; 7	—	0.04	8; 9	—	—	10	—	—
D0	2; 3	—	0.31	4; 5	—	0.08	6; 7	—	—	8; 9	0.02	—	10	0.02	—
D1	2; 3	—	0.96	4; 5	0.02	0.84	6; 7	0.09	0.67	8; 9	0.07	0.30	10	0.06	—
V0	2; 3	—	0.08	4; 5	—	—	6; 7	—	—	8; 9	—	—	10	—	—
V1	2; 3	—	**1.42**	4; 5	—	—	6; 7	—	—	8; 9	—	—	10	—	—
Averages		—	0.39		—	0.22		0.02	0.15		0.02	0.04		0.02	—

Source: Andre Paulino de Lima, 2019

Table 7 – Effect sizes obtained from linear regression models fit to the relative difference between exact and approximate values of potential surprise. In the column "Model", the model equation is defined: the factors *repmodel*, *distfn*, and *length* respectively correspond to item representation, distance function, and length of the sequence. R^2 represents the amount of variation that can be explained by the model (systematic variation).

Difference	Model	R^2	Difference	Model	R^2
DSpmin	repmodel	0.005	DSpmax	repmodel	0.005
DSpmin	distfn	0.010	DSpmax	distfn	0.005
DSpmin	length	0.010	DSpmax	length	0.003
DSpmin	repmodel * distfn	0.017	DSpmax	repmodel * distfn	0.018
DSpmin	repmodel * length	0.020	DSpmax	repmodel * length	0.011
DSpmin	distfn * length	0.026	DSpmax	distfn * length	0.011
DSpmin	repmodel * distfn * length	0.042	DSpmax	repmodel * distfn * length	0.031

Source: Andre Paulino de Lima, 2019

combined with each item. Two restrictions were imposed in the process that extracts item vectors from the dataset to ensure that the same set of items and set of ratings are used across all compatible configurations of representation model and distance function. This restriction aimed at eliminating variation in the results of an experiment that could arise owing to differences in input data. As for the quality of greedy approximations to potential surprise, the results obtained here agree with those obtained in Experiment 1. Although the results still do not support the general claim that a greedy algorithm will always achieve approximations as good as those obtained, it adds support to the idea that a local approach can obtain adequate approximations to the potential surprise.

4.3 Experiment 3: analysing change in surprise owing to a change in config

We now move on to our fourth research question, which inquires into the effects that the choices of item representation model and distance function may exert on the competence of an RS at embedding surprise in its recommendations. This section is organised as follows: the experiment design is described in Section 4.3.1; the experiment results and some of their salient patterns are presented in Section 4.3.2; and the statistical analysis of the results is developed in Section 4.3.3. In the latter, the prevalence of the salient patterns is evaluated at different subsets of the data. The discussion of the results is developed in Section 4.3.4, which also presents some hypotheses that integrate the patterns with their discrepancies in specific experimental conditions.

4.3.1 Experiment design

This experiment adopted a within-subjects, factorial design with two factors[12] (item representation and distance function). To control for variability, a controlled environment was created to carry out this experiment. The environment has three components: the experiment specification (config), a recommender system instance, and an evaluator. The config specifies: a) a recommendation algorithm, b) an item representation model, c) a distance function, and d) a sample of users. The recommender system produces a single recommendation list to each user in the sample by means of the recommendation algorithm specified in the config. The recommendation lists are sent to the evaluator, which estimates the system-level surprise by applying the proposed evaluation method (Section 3.5).

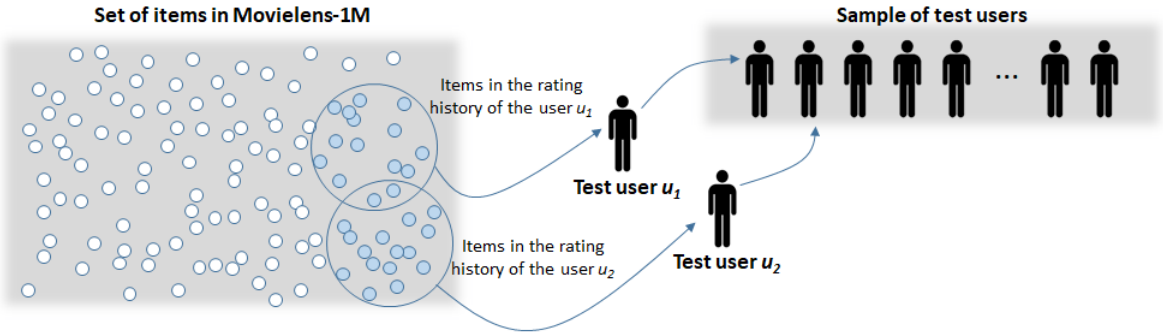
Method: apply the proposed evaluation method to a sample of users from the Movielens dataset; obtain system-level estimates of normalised surprise; aggregate the estimates to perform an analysis of variance.

Dataset: the extended MovieLens-1M Dataset was used to build the repository of the recommender system instance (see Section 4.2.1). A short note on the sample of users must be made here. This experiment was conducted as a computational simulation and, thus, did not involve human subjects. Instead, a user model was employed, as is

[12] Here, *factor*, *effect*, and *interaction* refer to their usage in experiment design (MONTGOMERY, 2017).

traditionally done in off-line experiments reported in the literature on RS[13]. The user model that was adopted employs the rating history of real users to simulate the previous experience of the test users that participate in the system evaluation, as illustrated in Figure 12. Each test user is identified by the "userID" of the user in the Movielens dataset from which the rating history was recovered. Given that the Movielens-1M dataset contains the rating history of 6.014 eligible users[14], a sample of users was drawn once and reused across all configs. It must be noted that the sample size ($n = 362$) was selected to obtain a 95% confidence level estimate of the mean normalised surprise with a margin of error of less than 5% (see Appendix B).

Figure 12 – Using ratings from real users in Movielens-1M to simulate test users.



Source: Andre Paulino de Lima, 2019

Procedure: given a target config, the evaluator assesses the average degree of normalised surprise of the recommendation lists produced by the recommender system to the users in the sample. The surprise obtained from each recommendation list is collected and stored, and the system-level estimate is taken as their average.

Variations: The Procedure was applied to the 56 configs that are illustrated in Figure 14, which were obtained by combining four recommendation algorithms (FP, N1, N2, or N3, see Section 2.7.2) with the compatible pairs of item representation (Model C, D, U, or V, see Section 2.5) and distance function (0 to 5, see Section 2.6). It must be noted that the surprise metric that is employed in each execution of the Procedure depends on the item representation and the distance function that are specified in the config, whereas the recommendation algorithm may follow a different pattern, as described in Table 2.

[13] This is the case of the one plus random evaluation methods described in Section 2.8.2.

[14] The dataset actually contains 6.040 users, of which 6.014 are eligible test users according to the one plus random evaluation method described in Algorithm 3. Seeking additional benefits that ensue from paring results obtained by both evaluation methods, the population of interest was restricted to this subset of users, assuming that the exclusion ($< 0.5\%$) did not severely bias the sampling procedure.

4.3.2 Experiment results

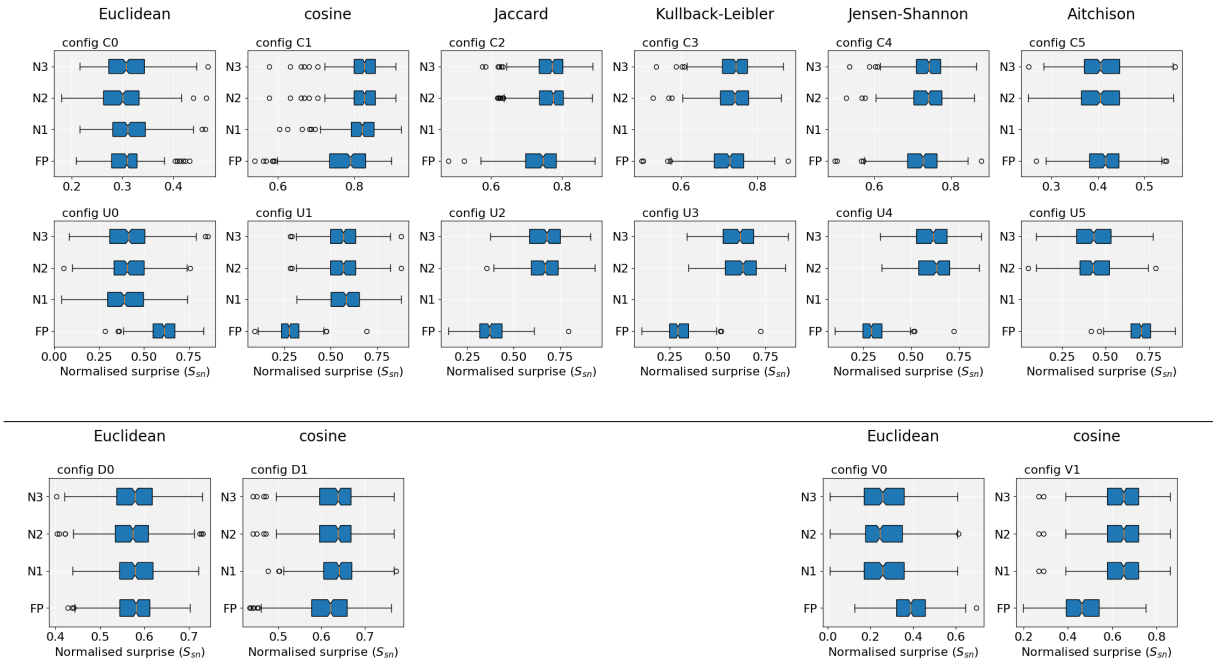
The results of the experiment are reported on Table 8 and, for convenience, also illustrated on Figure 13, in which the median and the interquartile range of the normalised surprise obtained from these configs are shown. Considering the confidence interval of each measurement, the results can be qualitatively described as follows: a) the mean normalised surprise values obtained from the neighbourhood-based algorithms (N1, N2, and N3) are predominantly indistinguishable from one another (exception in C0), and b) the mean normalised surprise obtained from the factorisation algorithm (FP) is predominantly lower than that obtained from the neighbourhood algorithms (exceptions in C0, C5, U0, U5, D0 and V0). In addition, another interesting pattern emerges when the results in Figure 13 are placed in juxtaposition with the results in Figure 11: the mean normalised surprise is predominantly higher than 0.5 for configs with negatively skewed distances and predominantly lower than that for configs with positively skewed distances. The first case includes 35 out of the 56 configs, but there are exceptions in FP for U0 and U5. The second case holds the remaining 21 configs, with exceptions in FP for U1, U2, U3, U4, and V1.

Table 8 – Mean normalised surprise obtained from the 56 configs in the experiment. Columns "mean" and "ci" (confidence interval) report on the results obtained from the recommendation algorithm indicated in the major column. Rows are listed in increasing order of skew and coefficient of variation (cv), and grouped as negatively skewed, fairly balanced ($|skew| < 0.5$), and positively skewed. Entries filled with a dash correspond to incompatible configurations. Intervals estimated at a 95% confidence level, with margin of error being inferior to 5%.

			Algorithm FP		Algorithm N1		Algorithm N2		Algorithm N3	
cfg	skew	cv	mean	ci	mean	ci	mean	ci	mean	ci
C3	-6.8	0.0361	0.722	0.716, 0.728	—	—	0.738	0.732, 0.743	0.740	0.734, 0.745
C1	-4.0	0.0155	0.776	0.769, 0.784	0.818	0.813, 0.822	0.823	0.818, 0.827	0.823	0.818, 0.827
U2	-2.6	0.0377	0.376	0.368, 0.385	—	—	0.663	0.652, 0.674	0.666	0.654, 0.678
U3	-2.3	0.0783	0.299	0.291, 0.308	—	—	0.621	0.610, 0.632	0.607	0.596, 0.618
U4	-1.9	0.0767	0.299	0.291, 0.308	—	—	0.621	0.610, 0.632	0.607	0.596, 0.618
C2	-1.8	0.0002	0.741	0.734, 0.747	—	—	0.767	0.762, 0.772	0.766	0.761, 0.771
C4	-1.6	0.0322	0.722	0.716, 0.728	—	—	0.738	0.732, 0.743	0.739	0.734, 0.745
U1	-1.6	0.0814	0.282	0.275, 0.290	0.577	0.566, 0.587	0.563	0.553, 0.574	0.563	0.553, 0.574
V1	-1.2	0.1340	0.465	0.454, 0.475	0.644	0.633, 0.654	0.644	0.633, 0.654	0.644	0.633, 0.654
D0	-0.3	0.1201	0.576	0.571, 0.581	0.582	0.576, 0.588	0.573	0.567, 0.579	0.579	0.573, 0.585
D1	-0.1	0.1208	0.615	0.609, 0.621	0.637	0.632, 0.643	0.634	0.628, 0.639	0.634	0.628, 0.639
C5	0.1	0.1752	0.414	0.409, 0.419	—	—	0.407	0.401, 0.413	0.410	0.404, 0.416
U5	0.3	0.4801	0.699	0.690, 0.707	—	—	0.435	0.422, 0.448	0.432	0.418, 0.446
C0	0.6	0.2678	0.306	0.302, 0.310	0.314	0.309, 0.319	0.299	0.294, 0.304	0.310	0.305, 0.315
U0	0.9	0.6355	0.607	0.597, 0.616	0.398	0.384, 0.412	0.418	0.405, 0.431	0.413	0.399, 0.427
V0	1.7	0.9597	0.391	0.381, 0.401	0.27	0.256, 0.283	0.266	0.254, 0.278	0.270	0.256, 0.283

Source: Andre Paulino de Lima, 2019

Figure 13 – The results obtained from the 56 configs in the experiment: the results from Model C are in the first row; from Model U in the second row; and from Models D and V in the third row, left and right, respectively. The notches around the median represent its confidence interval and were calculated using a Gaussian-based asymptotic approximation.



Source: Andre Paulino de Lima, 2019

4.3.3 Analysis of the results

Since each test user was assigned to all 56 conditions, the analyses were conducted according to a within-subjects design and, to promote statistical soundness, the configs were arranged into eight groups, as illustrated in Figure 14. In each group, the shaded area delimits a 2×6 (Groups 1, 5 and 7) or 4×2 (Groups 2, 4, 6 and 8) factorial design. Owing to particular incompatibilities that arise in Group 3, it becomes a subset of the Group 4 and, for that reason, it is not separately considered in the analyses[15].

[15] As the N1 algorithm uses item vectors from the **Q** matrix, which has negative elements, the following distance functions cannot be safely applied: Jaccard, Kullback-Leibler, Jensen-Shannon, and Aitchison.

Figure 14 – Composition of the factorial experiments. The codes indicate item representations (1^{st} character, organised in rows) and distance functions (2^{nd} character, organised in columns). Incompatible combinations are marked with a "-".

Group 1 (FP)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 3* (N1)

C0	C1	-	-	-	-
U0	U1	-	-	-	-
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 5 (N2)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 7 (N3)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 2 (FP)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 4 (N1)

C0	C1	-	-	-	-
U0	U1	-	-	-	-
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Group 6 (N2)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

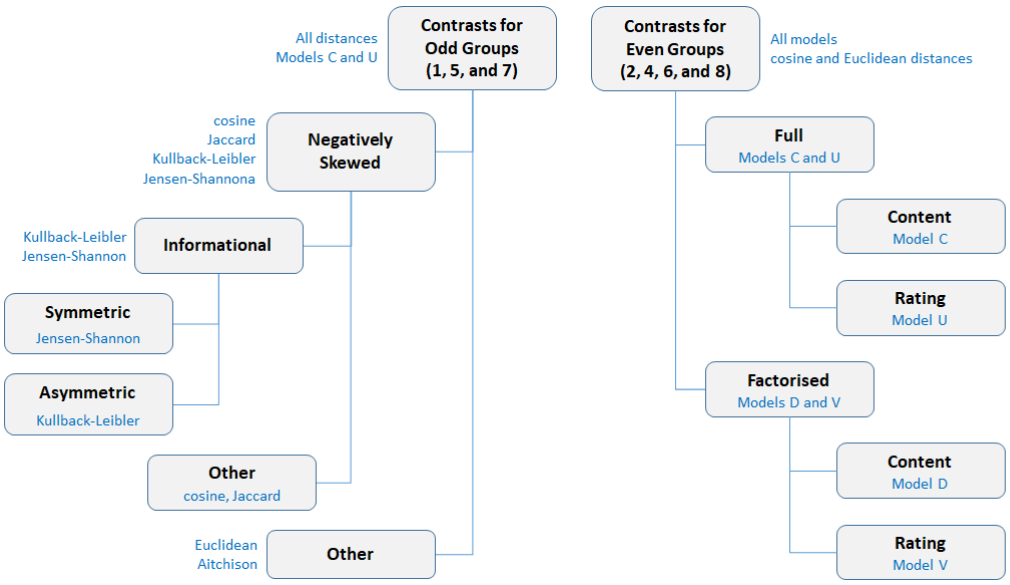
Group 8 (N3)

C0	C1	C2	C3	C4	C5
U0	U1	U2	U3	U4	U5
D0	D1	-	-	-	-
V0	V1	-	-	-	-

Source: Andre Paulino de Lima, 2019

A two-way repeated-measures ANOVA performed on the results from Groups 1, 5, and 7 allowed us to explore effects related to distance functions[16]. At a significance level of $p < 0.05$, all main effects and interactions were significant, so the contrasts that are illustrated in Figure 15a were applied to analyse effects at different resolutions of the data:

Figure 15 – Contrasts employed to analyse effects at different resolutions of the data. In the odd groups, the results are segregated by the skewness of the distance distribution, then by intuition, and lastly by symmetry. In the even groups, the results are separated into full and factorised, and then by source of data.



(a) Contrasts used in the 2x6 factorial designs.

(b) Contrasts used in the 4x2 factorial designs.

Source: Andre Paulino de Lima, 2019

[16] Results were resampled ($n = 20$, per config) and fitted to a multilevel linear model with contrasts.

Negatively skewed vs other: as seen in Figure 11, some distance functions produced negatively skewed distributions (cosine, Jaccard, Kullback-Leibler, and Jensen-Shannon) when applied to item vectors from Models C and U, whereas other distances produced non-negatively skewed distributions under the same conditions (Euclidean and Aitchison).

- Significance: the effect on the mean normalised surprise exerted by the negatively skewed distances (compared to the non-negatively skewed ones) was significant and large[17] for configs in all groups ($r > 0.5$, see Table 9).
- Pattern: as shown in Figure 16 (plots in the first column), the mean normalised surprise from configs with negatively skewed distributions (solid line) was predominantly higher than the mean normalised surprise from configs with non-negatively skewed distributions (dashed line).
  - Outlier 1: in Group 1, the mean normalised surprise from all configs under the Model U violates the pattern, as can be seen in Figure 16a.

Informational vs other: changing the resolution to focus on the configs with negatively skewed distributions, a contrast between distances with informational and non-informational intuition was set.

- Significance: the effect on the mean normalised surprise exerted by the informational distances (compared to non-informational ones) was predominantly non-significant.
  - Exception 1: for Group 5, in which the effect was significant and moderate ($r \in [0.1, 0.5]$, see Table 9). As illustrated by Figure 16e, although the mean normalised surprise from all configs were comparable under Model U, they differed under Model C: the mean normalised surprise from configs with informational distances (solid line) was lower than the mean normalised surprise from configs with non-informational distances (dashed line).

Symmetric vs asymmetric: to focus on the informational divergences, a contrast was set between symmetric (Jensen-Shannon) and asymmetric (Kullback-Leibler) functions.

- Significance: the effect on the mean normalised surprise of the symmetric distance (compared to the asymmetric distance) was non-significant, as reported in Table 9.

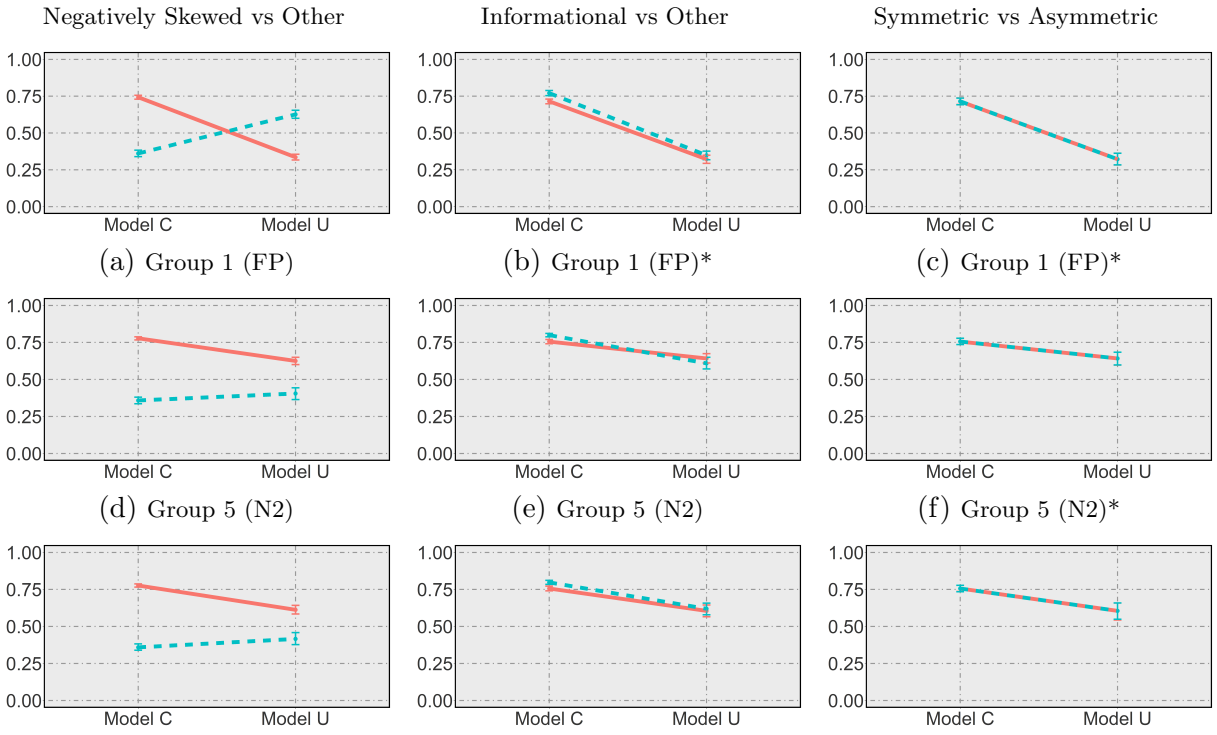
[17] The terms used to qualify effect sizes (small, medium, and large) follow definitions in (ELLIS, 2010).

Table 9 – Significance and effects sizes of the contrasts applied to the results from each group. Results were considered significant at $p < 0.05$. Effect sizes were reported in the r scale. In the last column, the direction of change is described as moving from the right to the left condition of the contrast. For example, in the "Negatively-skewed vs other" contrast in Group 1 (first row in the table), the increase in mean normalised surprise in Model C occurs if one moves from the "other" condition to the "Negatively-skewed" condition.

Contrast	Grp	b	df	t(df)	p-value	Effect	Change in the mean surprise
Negatively-skewed vs other	1(FP)	0.112	190	41.64	< 0.001	0.949	Increase in Model C, decrease in Model U.
Negatively-skewed vs other	5(N2)	0.033	190	8.15	< 0.001	0.509	Increase in all models.
Negatively-skewed vs other	7(N3)	0.037	190	8.80	< 0.001	0.538	Increase in all models.
Informational vs other	1(FP)	-0.007	190	-1.57	0.117	—	Change is not significant.
Informational vs other	5(N2)	-0.019	190	-2.71	0.007	0.193	Decrease in Model C, not significant in Model U.
Informational vs other	7(N3)	-0.007	190	-0.94	0.350	—	Change is not significant.
Symmetric vs asymmetric	1(FP)	0.000	190	0.00	0.998	—	Change is not significant.
Symmetric vs asymmetric	5(N2)	0.000	190	0.00	0.999	—	Change is not significant.
Symmetric vs asymmetric	7(N3)	0.000	190	0.00	0.998	—	Change is not significant.
Full vs Factorised	2(FP)	0.009	76	1.95	0.055	—	Change is not significant.
Full vs Factorised	4(N1)	0.035	76	5.00	< 0.001	0.498	Increase in Euclidean distance, decrease in cosine.
Full vs Factorised	6(N2)	0.028	76	3.86	< 0.001	0.405	Increase in Euclidean distance, decrease in cosine.
Full vs Factorised	8(N3)	0.029	76	3.98	< 0.001	0.415	Increase in Euclidean distance, decrease in cosine.
Content vs Rating (Full)	2(FP)	0.191	76	27.83	< 0.001	0.954	Increase in cosine distance, decrease in Euclidean.
Content vs Rating (Full)	4(N1)	0.074	76	7.58	< 0.001	0.656	Increase in cosine distance, decrease in Euclidean.
Content vs Rating (Full)	6(N2)	0.091	76	9.05	< 0.001	0.702	Increase in cosine distance, decrease in Euclidean.
Content vs Rating (Full)	8(N3)	0.089	76	8.71	< 0.001	0.707	Increase in cosine distance, decrease in Euclidean.
Content vs Rating (Fact.)	2(FP)	-0.012	76	-1.69	0.096	—	Change is not significant.
Content vs Rating (Fact.)	4(N1)	-0.075	76	-7.67	< 0.001	0.661	Increase in Euclidean distance, decrease in cosine.
Content vs Rating (Fact.)	6(N2)	-0.084	76	-8.37	< 0.001	0.693	Increase in Euclidean distance, decrease in cosine.
Content vs Rating (Fact.)	8(N3)	-0.078	76	-7.65	< 0.001	0.660	Increase in Euclidean distance, decrease in cosine.

Source: Andre Paulino de Lima, 2019

Figure 16 – Interaction plots for contrasts applied to Groups 1, 5, and 7. Representation models in the abscissa; ordinate in normalised surprise scale. In the first column, there are the plots from the "Negatively skewed (solid line) vs other (dashed)" contrast; in the second column, the plots from the "Informational (solid) vs other (dashed)" contrast, and in the third column, the plots from the "Symmetric (solid) vs Asymmetric (dashed)" contrast. An asterisk in the caption indicates that the plot corresponds to a non-significant interaction.



(a) Group 1 (FP) (b) Group 1 (FP)* (c) Group 1 (FP)*

(d) Group 5 (N2) (e) Group 5 (N2) (f) Group 5 (N2)*

(g) Group 7 (N3) (h) Group 7 (N3)* (i) Group 7 (N3)*

Source: Andre Paulino de Lima, 2019

A similar analysis performed on the results from Groups 2, 4, 6, and 8 allowed us to explore effects related to item representation. At the confidence level of $p < 0.05$, all main effects and interactions were significant, and the contrasts in Figure 15b revealed that: *Full vs Factorised*: as shown in Figure 11, the distributions produced by the Euclidean distance differ substantively whether they are extracted from full or factorised models, and the same can be stated about the distributions produced by the cosine distance. More specifically, factorised models that use SVD (Models V0 and V1) have a higher skewness and coefficient of variation than those from their respective full models (Models U0 and U1), while factorised models that use SGD (Models D0 and D1) have skewness near zero and similar coefficients of variation, despite of the differences between their respective full models (Models C0 and C1).

- Significance: the effect on the mean normalised surprise exerted by full models (compared to factorised models) was predominantly significant with moderate to large effects for configs in Groups 4, 6, and 8 ($r > 0.4$, see Table 9).
  - Exception 2: the effect was marginally non-significant for Group 2 ($p = 0.055$).
- Pattern: as illustrated in Figure 17 (plots in the first column), for groups in which the effect was significant, the mean normalised surprise from configs with full models (solid line) was lower than the mean normalised surprise from configs with factorised models (dashed line) under the cosine, and the opposite under the Euclidean distance.

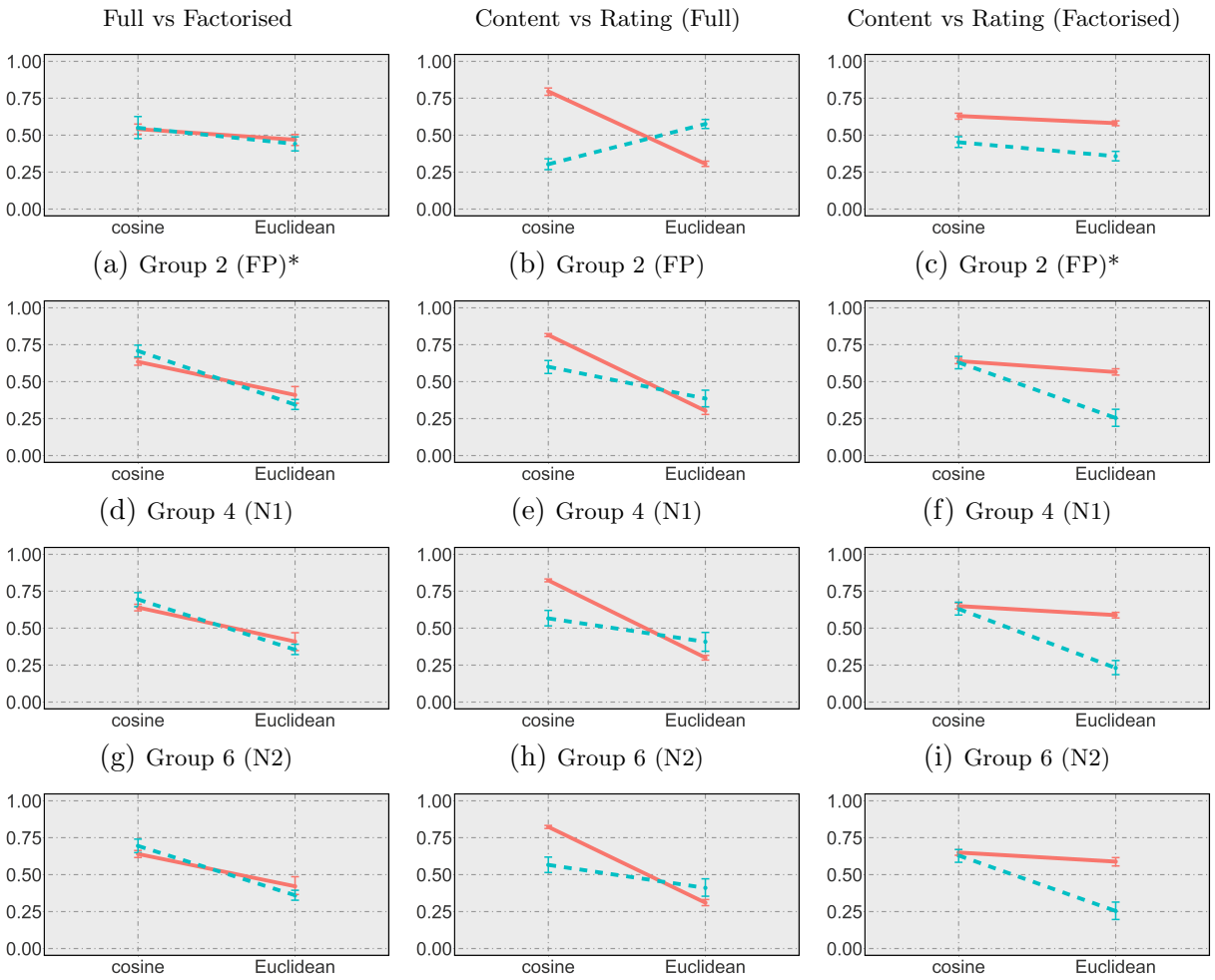
Content vs Ratings (full models): to focus on the configs with full models, a contrast between content-based (Model C) and rating-based (Model U) models was set.

- Significance: the effect on the mean normalised surprise exerted by content data (compared to rating data) was significant and large for configs in all groups ($r > 0.5$).
- Pattern: as illustrated in Figure 17 (plots in the second column), the mean normalised surprise from configs with content data (solid line) is higher under the cosine distance than under the Euclidean distance; the same pattern is obtained from the configs with rating data (dashed line).
  - Outlier 2: in Group 2, the mean normalised surprise from the config with rating data (dashed line) is higher under the Euclidean distance than under the cosine.

Content vs Ratings (factorised models): to focus on factorised models, a new contrast between content-based (Model D) and rating-based (Model V) models was set.

- Significance: the effect on the mean normalised surprise exerted by content data (compared to rating data) was significant and large for configs in Groups 4, 6, and 8 ($r > 0.5$, see Table 9).
  - Exception 3: the effect was non-significant for Group 2 ($p = 0.096$).
- Pattern: as shown in Figure 17 (plots in the third column), for the groups in which the effect was significant, the mean normalised surprise from all configs were comparable under the cosine distance (in the discussion, this fact is referred to as Coincidence 1), but differed under the Euclidean distance, as the mean normalised surprise from the config with content data (config D1, solid line) was higher than the mean normalised surprise from the config with rating data (config V1, dashed line).

Figure 17 – Interaction plots for contrasts applied in Groups 2, 4, 6, and 8. Distances represented in the abscissa; ordinate in normalised surprise scale. In the first column, there are the plots from the "Full (solid line) vs Factorised (dashed line)" contrast; in the second column, the plots from the "Content (solid) vs Rating (dashed) within full models" contrast, and in the third column, the plots from the "Content (solid) vs Rating (dashed) within factorised models" contrast. An asterisk in the caption indicates that the plot corresponds to a non-significant interaction.



(a) Group 2 (FP)* (b) Group 2 (FP) (c) Group 2 (FP)*

(d) Group 4 (N1) (e) Group 4 (N1) (f) Group 4 (N1)

(g) Group 6 (N2) (h) Group 6 (N2) (i) Group 6 (N2)

(j) Group 8 (N3) (k) Group 8 (N3) (l) Group 8 (N3)

Source: Andre Paulino de Lima, 2019

4.3.4 Hypotheses to integrate the found patterns and discrepancies

The evidences obtained from this experiment support the claim that choices of item representation and the distance function employed in item comparison exert a significant effect on the capacity of a recommender system at embedding the available surprise in its recommendations. This finding agrees with a notion of surprise that is defined in terms of the dissimilarity between item vectors, as formalised in Equation 11. In most conditions that were explored in the experiment, the size of the effect varied from medium to large, which answers our last research question (RQ4) in a qualitatively way.

In addition, the evidences suggest that a) the more negatively skewed is the pairwise distance distribution, the higher is the mean normalised surprise, and b) the mean normalised surprise obtained from the factorisation algorithm (FP) is predominantly lower than that obtained from the neighbourhood algorithms. However, these patterns found discrepancies in the results that were presented Section 4.3.3, as is summarised in Table 10. In the following sections, we try to make sense of these discrepancies and integrate them into a theory that could explain why they occur in some conditions, and not in others.

Table 10 – Summary of patterns and discrepancies found in the analysis. In the column "Pattern", it is described a pattern that is salient or expected at a given level of the results. For example, in the first row, $\bar{S}_{sn}|_{N_i} \equiv \bar{S}_{sn}|_{N_j}$ means that the mean normalised surprise values obtained from any two neighbourhood-based algorithms, N_i and N_j, are predominantly indistinguishable from one another. In the columns "Scope", "Grp" (group), and "Lev" (level), it is indicated the level of the results in which the pattern holds. The exceptional instances are described in the last column.

Reference	Pattern	Scope	Grp	Lev	Exceptional Instances
Salient 1	$\bar{S}_{sn}\|_{N_i} \equiv \bar{S}_{sn}\|_{N_j}$	All	—	—	$\bar{S}_{sn}\|_{N2} < \bar{S}_{sn}\|_{N_j}$ for C0.
Salient 2	$\bar{S}_{sn}\|_{FP} < \bar{S}_{sn}\|_{N_i}$	All	—	—	$\bar{S}_{sn}\|_{FP} \equiv \bar{S}_{sn}\|_{N_i}$ for C0,C5,D0; $\bar{S}_{sn}\|_{FP} > \bar{S}_{sn}\|_{N_i}$ for U0,U5,V0.
Salient 3	$\bar{S}_{sn}\|_{skew-} > 0.5$ $\bar{S}_{sn}\|_{skew+} < 0.5$	All	—	—	$\bar{S}_{sn}\|_{skew-} < 0.5$ in FP for U1,U2,U3,U4,V1; $\bar{S}_{sn}\|_{skew+} > 0.5$ in FP for U0,U5.
Outlier 1	$\bar{S}_{sn}\|_{skew-} > \bar{S}_{sn}\|_{skew+}$	Odd	G1	1	$\bar{S}_{sn}\|_{skew-} < \bar{S}_{sn}\|_{skew+}$ in FP for U*.
Outlier 2	$\bar{S}_{sn}\|_{skew-} > \bar{S}_{sn}\|_{skew+}$	Even	G2	2	$\bar{S}_{sn}\|_{skew-} < \bar{S}_{sn}\|_{skew+}$ in FP for U0, U1.
Coincid. 1	$\neg(\bar{S}_{sn}\|_{D_i} \equiv \bar{S}_{sn}\|_{V_i})$	Even	All	3	$\bar{S}_{sn}\|_{D1} \equiv \bar{S}_{sn}\|_{V1}$ in N*.
Exception 1	Non-significant effect	Odd	G5	2	$\bar{S}_{sn}\|_{(kl+js)} < \bar{S}_{sn}\|_{(cos+jac)}$.
Exception 2	Significant effect	Even	G2	1	$\bar{S}_{sn}\|_{full} \equiv \bar{S}_{sn}\|_{factorised}$
Exception 3	Significant effect	Even	G2	3	$\bar{S}_{sn}\|_{content} \equiv \bar{S}_{sn}\|_{rating}$

Source: Andre Paulino de Lima, 2019

A purported link between surprise and the skewness of the pairwise distance distribution

In this section, we analyse two instances that support the correspondence between a negatively skewed distance distribution and a higher degree of surprise (configs U0N3 and U1N3), and two instances that do not support the pattern (configs U0FP and U1FP). We then argue that the use of factorisation to produce recommendations can explain the following discrepancies: Outlier 1, Outlier 2, Salient 2, and Salient 3 (see Table 10).

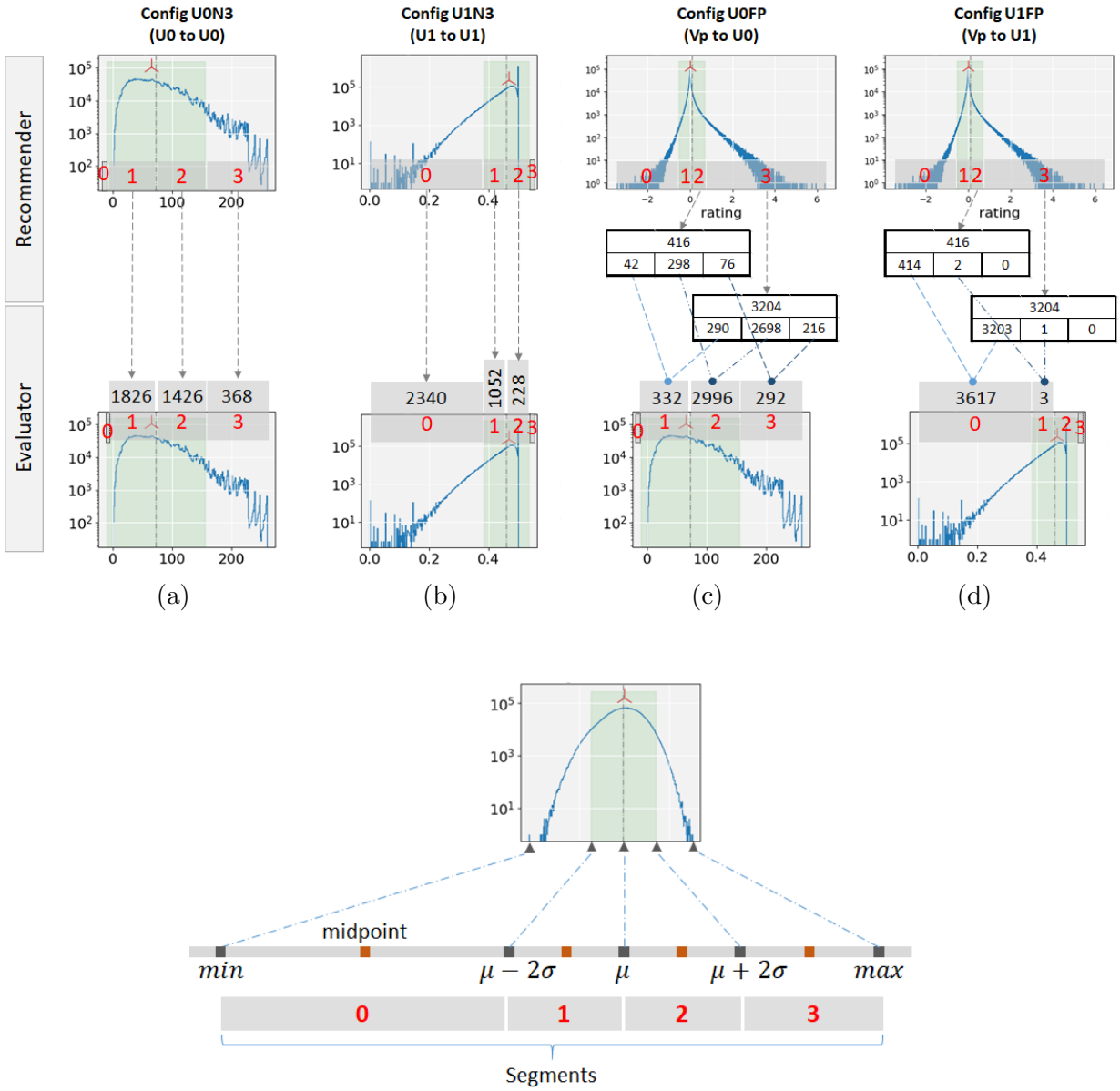
In config U0N3, recommendation lists are produced by a neighbourhood-based recommender operating under the representation model U0. The recommender applies the metamodel to each candidate item in a list that is combined with a test user (list $L1$ in line 8 of Algorithm 4). In turn, the metamodel estimates the rating of each candidate item by solving a weighted k-nearest neighbours interpolation. As a result, the candidate items that are closer to items highly rated by the user are more prone to be selected and make into the recommendation list produced to that user (Algorithm 4, line 11).

We use the config U0N3 as a background to present an explicative model for the association between surprise and the skewness of its related distance distribution. The model corresponds to a mapping between two distributions, as illustrated in Figure 18a. At the top of the figure, there is the distance distribution from the config employed by the recommender (U0), and, at the bottom, there is the distance distribution from the config employed by the evaluator (U0); for convenience of the analysis, both distributions were divided into four segments[18]. In the middle, there is a mapping between the segments from the source distribution (used by the recommender) to the target distribution (used by the evaluator). The mapping reflects the number of recommended items that fits the descriptions of a source segment A and a target segment B, and is build as follows:

- As mentioned earlier, 362 test users were assigned to each config, and a single recommendation list was generated to each user. As each list contained 10 items, the recommender selected 3,620 items to compose the recommendation lists.
- Using the config employed by the recommender, the minimum distance between a recommended item and the set of items known to its target user is assessed, and this value is used to determine its source segment. The idea is to gain some insight

[18] Distributions were segmented as described in Figure 18e.

Figure 18 – Recommended items are mapped according to the distributions employed by the recommender and the evaluator. Recommendation lists were decomposed into recommended items, which were mapped to the segment of the distribution from which they were drawn by the recommender, and also to the distance distribution employed by the evaluator to assess surprise. The counts reflect the number of recommended items that flow from the source segment to their target segment.



(e) Distributions were divided into 4 segments: segment 0 covers the interval between the minimum value up to $(\mu - 2\sigma)$, segment 1 covers $[\mu - 2\sigma, \mu)$, segment 2 covers $[\mu, \mu + 2\sigma)$, and segment 3 covers from there up to the maximum value. This choice of segmentation has some important features: a) the structure of the segments is qualitatively discernible for negatively and positively skewed distributions: in the former, the highest segment is empty (see 18b), and in the latter, the first segment is empty (see 18a); b) the number of segments is small, but good approximations to non-normalised surprise can still be obtained by means of interpolation.

Source: Andre Paulino de Lima, 2019

about the existence of regions of the item representation space that are more prolific in producing recommendations, from a user-centric perspective.

- Using the config employed by the evaluator, the minimum distance between a recommended item and the set of items known to its target user is assessed, and this value is used to determine its target segment. The idea here is to obtain a rough estimate of the contribution of the recommended item to the system-level surprise.

For example, the arrow that links the source segment 1 to the target segment 1 in Figure 18a indicates that a) the minimum distance between a recommended item and an item known to the user is within $[\mu - 2\sigma, \mu]$ for 1,826 of the 3,620 recommended items; and b) since both recommender and evaluator use the same config, then the distance between any of those 1,826 items and an item known to its target user can be roughly estimated as the midpoint of the segment 1 (this point is elaborated ahead). Considering the remaining arrows, it can be seen that the number of recommended items that keep a distance below the distribution mean from an item known to its target user is comparable to the number of items that keep a distance larger than that ($1,826 \approx 1,426 + 368 = 1,794$). It must be noted that the recommended items are massively concentrated within $[\mu - 2\sigma, \mu + 2\sigma]$, and that no item from the segment 0 was selected because the 2σ interval around the mean left this segment with a null fraction of the mass of the distribution. This result is, at least in part, an effect of the distance distribution being positively skewed (+0.9) and having a relatively large coefficient of variation (0.5728).

Considered as a whole, the target mapping can be used to obtain an estimate of the mean (non-normalised) surprise per item by means of weighted interpolation:

$$\begin{aligned}\bar{S}_i|_{U0}^{U0} &= \frac{1}{3,260}\left[1,826\,\frac{(\mu - 2\sigma) + \mu}{2} + 1,426\,\frac{\mu + (\mu + 2\sigma)}{2} + 368\,\frac{(\mu + 2\sigma) + max}{2}\right] \\ &= \frac{1}{3,260}\left[1,826 \times 36.3 + 1,347 \times 114.1 + 368 \times 207.6\right] = 84.4.\end{aligned}$$

An interesting pattern in the results is that the system-level estimate of normalised surprise ($\hat{S}'_{sn}$) for most configs can be approximated by a low fidelity model ($\bar{S}_{sn}$) that employs non-normalised, item-level surprise ($\bar{S}_i$) as a proxy for surprise of a sequence (S_s), and extremes of surprise of an item (lb and ub) as proxies of potential surprise

($\hat{S}'_{pmin}$ and $\hat{S}'_{pmax}$, respectively)[19]. For example, taking the outer midpoints to the dominant segments 1 and 2 (midpoints of segments 0 and 3) as estimates for lb and ub, we have:

$$\bar{S}_{sn}|_{U0}^{U0} = \frac{\bar{S}_i - lb}{ub - lb} = \frac{84.4 - 0.0}{207.6 - 0.0} = 0.406 \approx 0.413 = \hat{S}_{sn}|_{U0}^{U0} \text{ (see Table 8, config U0N3).}$$

An approximation as good as this could also be obtained by taking ub and lb as the minimum and maximum pairwise distances ($lb = 0.0, ub = 259.5$).

We now move on to analyse how the low fidelity model fits to a config with a negatively skewed distance distribution. Under the config U1N3, recommendation lists were produced according to the representation model U1 (item vectors from the user-item matrix are compared by applying the cosine distance). The distance distribution has a skew of -1.6 and a coefficient of variation of 0.0829, and is illustrated in Figure 18b. From the recommended items, 2,340 items were drawn from the segment 0; 1,052 from segment 1, and 228 items from segment 2. No item was drawn from the segment 3 for the same reason that segment 0 was null in the config U0N3. Since the evaluator also used U1 to assess surprise, an estimate for the system-level surprise can be obtained by taking the minimum and maximum pairwise distances as lb and ub in the low fidelity model:

$$\begin{aligned}
\bar{S}_i|_{U1}^{U1} &= \frac{1}{3,260}\left[2,340\,\frac{min + (\mu - 2\sigma)}{2} + 1,052\,\frac{(\mu - 2\sigma) + \mu}{2} + 228\,\frac{\mu + (\mu + 2\sigma)}{2}\right] \\
&= \frac{1}{3,260}\left[2,340 \times 0.192 + 1,052 \times 0.422 + 228 \times 0.480\right] = 0.277; \\
\bar{S}_{sn}|_{U1}^{U1} &= \frac{\bar{S}_i - lb}{ub - lb} = \frac{0.277 - 0.0}{0.500 - 0.0} = 0.554 \approx 0.563 = \hat{S}_{sn}|_{U1}^{U1}.
\end{aligned}$$

A similar approximation ($|\hat{S}_{sn} - \bar{S}_{sn}| < 0.1$) could also be obtained by taking lb and ub as the outer midpoints to the dominant segments (min and the midpoint of segment 2).

The two configs analysed, U0N3 and U1N3, support the pattern that configs with negatively skewed distance distributions obtain a higher degree of surprise than configs with a non-negatively skewed distribution. This pattern reflects the mapping of recommended items into the representation space in which surprise is assessed. In the first config, which is positively skewed, the mean item surprise ($\bar{S}_i|_{U0}^{U0} = 84.4$) was below the middle of the range of variation where most of the mass of the distance distribution was located ($\frac{1}{2}(ub - lb) = 103.8$); in the second config, which is negatively skewed, the mean item surprise ($\bar{S}_i|_{U1}^{U1} = 0.277$) was above the middle of the range of variation ($\frac{1}{2}(ub - lb) = 0.25$).

[19] The inequality $|\hat{S}'_{sn} - \bar{S}_{sn}| < 0.1$ holds for 42 out of the 56 configs.

However, the pattern was broken when a factorisation-based recommender was employed. The first thing to note is that neighbourhood-based recommendation and surprise assessment have two common fundamental concepts: representation and comparison. Both require that items be represented as vectors, and a function to evaluate their degree of (dis)similarity. As mentioned before, candidate items are more prone to make into a recommendation list if they are close to items that have been highly rated by the user. In contrast, factorisation-based algorithms select items with a higher predicted score, and this score is computed as the inner product between a user vector and an item vector from the latent features space obtained by factorising the user-item matrix. In Figure 18c, the distribution sampled by the recommender is not a distance distribution; instead, it is an inner product distribution, obtained by computing the histogram of the predicted scores from all pairs of items and test users. The distribution is substantially skewed (+4.8) and spread ($cv = 3.530$), and, as before, the distribution was divided in the four segments described in Figure 18e. From the recommended items, 416 items were drawn from segment 2, and the remaining 3,204 from segment 3. No items were selected from segments 0 and 1 because their predicted score is negative, making them irremediably noncompetitive. As the evaluator used U0 to assess surprise, we use the same estimates for lb and ub that were employed for config U0N3 to obtain an estimate from the low fidelity model:

$$\bar{S}_i|_{U0}^{Vp} = \frac{1}{3,260}\Big[332 \times 36.3 + 2,996 \times 114.1 + 292 \times 207.6\Big] = 114.5.$$
$$\bar{S}_{sn}|_{U0}^{Vp} = \frac{\bar{S}_i - lb}{ub - lb} = \frac{114.5 - 0.0}{207.6 - 0.0} = 0.552 \approx 0.607 = \hat{S}_{sn}|_{U0}^{Vp}.$$

The increase in $\bar{S}_{sn}$ with respect to config U0N3 was brought about by the difference in the mapping of recommended items to the representation space in which surprise is assessed: recommended items were massively drawn the segment 2, unlike the mapping for U0N3, which shows a more balanced allocation.

In Figure 18d, the mapping for the config U1FP is shown. As the evaluator used U1 to assess surprise, we use the same estimates for lb and ub that were employed for config U1N3 to obtain an estimate from the low fidelity model:

$$\bar{S}_i|_{U1}^{Vp} = \frac{1}{3,260}\Big[3,617 \times 0.192 + 3 \times 0.422\Big] = 0.192;$$
$$\bar{S}_{sn}|_{U1}^{Vp} = \frac{\bar{S}_i - lb}{ub - lb} = \frac{0.192 - 0.0}{0.500 - 0.0} = 0.384 \approx 0.282 = \hat{S}_{sn}|_{U1}^{Vp}.$$

The sharp decrease in $\bar{S}_{sn}$ with respect to config U1N3 was brought about by the difference in the mapping: recommended items were massively drawn from the segment 0, unlike the mapping for U1N3, which shows a more balanced allocation.

We argue that a possible explanation for the discrepancies that were pointed out in the configs U0FP and U1FP, as well as in other exceptional instances listed in Table 10 (Salient 2, Salient 3, Outlier 1 and Outlier 2), is related to properties of the inner product:

- PureSVD selects items with highest estimated scores. The score is computed as the inner product between a user vector and an item vector, which are extracted from the latent features space obtained by applying SVD to the user-item matrix (Model V). Item vectors are extracted from the $\mathbf{Q}$ matrix, and user vectors are obtained by multiplying the raw user ratings vector and the $\mathbf{Q}$ matrix.

- Let $\mathbf{r}_u$ be a fixed user vector. Assume the following intuitive statements as true:

  a) the smaller the angle between $\mathbf{r}_u$ and an arbitrary item vector $\mathbf{q}_i$, the higher their cosine similarity, and the higher the score obtained by $\langle \mathbf{r}_u, \mathbf{q}_i \rangle$.

  b) the larger the magnitude of $\mathbf{q}_i$, measured as $||\mathbf{q}_i||$, the higher the score resulting from $\langle \mathbf{r}_u, \mathbf{q}_i \rangle$. Joining these intuitions together, we have:

$$\frac{\langle \mathbf{r}_u, \mathbf{q}_i \rangle}{||\mathbf{r}_u||\,||\mathbf{q}_i||} > \frac{\langle \mathbf{r}_u, \mathbf{q}_j \rangle}{||\mathbf{r}_u||\,||\mathbf{q}_j||} \wedge ||\mathbf{q}_i|| > ||\mathbf{q}_j|| \implies \langle \mathbf{r}_u, \mathbf{q}_i \rangle > \langle \mathbf{r}_u, \mathbf{q}_j \rangle.$$

- Let $\mathbf{v}_i$ correspond to the item vector of i, according to Model U. Thus, $\mathbf{q}_i$ is a lower dimensional representation of $\mathbf{v}_i$. Assume the following intuitive statements as true:

  a) if $\mathbf{q}_i$ keeps a relatively small angle with $\mathbf{r}_u$, then $\mathbf{v}_i$ is expected to keep a relatively small angle with some item vector $\mathbf{v}_j$, $j \in E_u$;

  b) If $\mathbf{q}_i$ has a relatively large magnitude, then $\mathbf{v}_i$ is expected to keep a relatively large (Euclidean) distance from item vectors $\mathbf{v}_j$, $j \in E_u$.

Under these conditions, PureSVD is expected to recommend items that keep a relatively small cosine distance from items known to the user. As a result, a larger share of recommended items would be drawn from the segments under the distribution mean. This is in agreement with the mapping obtained for the config U1FP (Figure 18d). In addition,

PureSVD is also expected to select items that keep a relatively large Euclidean distance from the items known to the user, and, as a result, a larger share of recommended items would be drawn from segments above the distribution mean. This is in agreement with the mapping for the config U0FP (Figure 18c). However, as this explanation is limited to configs with Models U or V, it remains unclear if it can be extended to the configs C0, C5, and D0, listed in Salient 2.

A purported link between surprise and matrix factorisation

It is believed that neighbourhood-based algorithms should produce recommendations with a higher degree of surprise than algorithms that explore matrix factorisation (KAMINSKAS; BRIDGE, 2014). In fact, the idea that some recommendation algorithms embed less surprise than others is explored by the metrics of surprise that use an external system (PPM) to produce recommendations that are used as baselines for surprise (MURAKAMI; MORI; ORIHARA, 2008; GE; DELGADO-BATTENFELD; JANNACH, 2010; ADAMOPOULOS; TUZHILIN, 2011). However, the causes of this difference remain largely unexplored to the best of our knowledge. In this experiment, some results that contradict this expected pattern were obtained; they are summarised in Table 10, under the pattern "Salient 2". As seen in the previous topic, the explanative model attributes the higher surprise value obtained by factorisation as an effect of combining a recommender based on PureSVD and using the Euclidean distance to assess surprise. Once more, the explanative model covers the exceptional instances related to the Models U and V, but it remains unclear if it can be extended to the exceptional instances with Models C or D.

4.4 Experiment 4: comparing current and proposed evaluation methods

As described in Section 2.8.2, the one plus random method was adapted to evaluate system-level surprise in (KAMINSKAS; BRIDGE, 2014) and, in addition, this work has also defined a metric for surprise based on the Jaccard distance that is used in tandem with the adapted method. More important, the work reports on the results obtained from applying the method to the Movielens 1M dataset. For these reasons, we adopted it as a baseline, but it is important to make it clear that, in the mentioned work, the method was applied

to items represented as sets of labels that differ from those that are original to the dataset, and the extended dataset was not made publicly available.

Method: apply the one plus random method and the proposed method to a sample of users from the Movielens dataset; obtain values for surprise, recall, and execution times; compare results obtained to the baseline.

Dataset: the extended Movielens 1M dataset described in Section 4.2.1 was used. A random sample of 50 eligible users[20] was drawn once and reused by both methods. To set up experimental conditions comparable to those adopted in the baseline, item vectors were extracted from the textual content combined with each item and surprise was evaluated using the Jaccard distance. This setting is similar to the config C2 that was explored in the previous experiment. It must be noted that, in this config, the average number of tokens per item in the mapping D is 35, whereas the baseline reports the average of 81 labels per item.

Procedure: given a metamodel, a merged version of the one plus random methods described in Algorithms 2 and 3 is used to estimate system-level recall and non-normalised surprise. Analogously, the proposed method described in Algorithm 4 is used to estimate the system-level normalised and non-normalised surprise. Values for recall and surprise are collected and recorded for both methods, along with their execution times[21].

Variations: the Procedure was applied to two metamodels presented in Section 2.7: the factorisation-based PureSVD (FP) metamodel and one of the neighbourhood-based metamodels (N2), which respectively correspond to the MF (matrix factorisation) and the IB (item-based) algorithms that were explored in the baseline.

Results and discussion: the results are summarised in Table 11. The values in the S_{cont} columns (content-based surprise, reported for the one plus random and the baseline) were computed using Equation 10, while the values in the $\bar{S}_s$ column were calculated from Equation 35 divided by $N = 10$. The values in S_{sn} were obtained from Equation 38. In

[20] A user is eligible if they satisfy the requirements to be a test user of both evaluation methods.

[21] Execution times collected correspond to system plus user CPU time and do not account for sleep time. As this process is memory-intensive, the measurement was done with garbage collection enabled.

the following sections, we compare the estimates for surprise and the execution times that were obtained.

A comparative analysis of the resulting surprise

In comparing the results from the neighbourhood-based algorithm (N2) with its respective baseline (IB), it can be seen that the estimates of non-normalised surprise obtained from the one plus random (S_{cont}) and from the proposed method ($\bar{S}_s$) are just above the reference value ($S_{cont} \in [0.930, 0.941]; \bar{S}_s \in [0.932, 0.941]; S_{cont}, \bar{S}_s > 0.927$). There is also a difference in the recall rate, which is lower in the one plus random if compared to the baseline ($recall \in [0.000, 0.059], recall < 0.065$). Regarding the results from the factorisation-based algorithm (FP), it can be seen that the estimate of non-normalised surprise obtained from the one plus random (S_{cont}) is well below the reference value ($S_{cont} \in [0.856, 0.881]; S_{cont} < 0.915$), whereas the recall rate is much higher if compared to the baseline ($recall \in [0.527, 0.793], recall > 0.334$). In contrast, the non-normalised surprise obtained from the proposed method ($\bar{S}_s$) is just above its reference value ($\bar{S}_s \in [0.926, 0.932]; \bar{S}_s > 0.915$).

Two factors that can possibly explain the deviations of the non-normalised surprise and the recall rate from their respective references in the baseline are a) the use of different content data, and b) differences in the partitioning of the rating data. Given that we adopted the representation model C2, content data determine the item vectors. If the use of less descriptive content data leads to lower performance in recall, then a link between recall and non-normalised surprise in Algorithm 3 can explain the deviations in N2:

- The item i^* is an item highly rated by the user (line 5), which is included in the list of candidate items (line 7). If this item makes it into the recommendation list, it has the effect of decreasing the average surprise because $i^* \in E_u \implies S_i(i^*, E_u) = 0$.
- Therefore, the more frequently i^* makes it into the recommendation list, the higher the recall rate, and the lower the expected surprise.

In N2, the recall is just below its reference value, so the non-normalised surprise is expected to be higher, and this is in agreement with the evidences. Analogously, the lower non-normalised surprise for FP assessed by means of the one plus random is related its higher recall rate. In this case, one reason that can explain the higher recall rate is

Table 11 – Comparison of the results obtained from the one plus random and the proposed method applied to our extended Movielens 1M dataset. Algorithms indicated as MF and FP are factorisation-based algorithms and correspond to the PureSVD, with $f = 50$ dimensions, whereas IB and N2 are neighbourhood-based algorithms, with $k = 50$. Values reported as averages, except Time (total, in seconds), recall and surprise (15% confidence interval at 95% confidence level).

Baseline			One plus random				Proposed		
Alg.	S_{cont}	Recall	Alg.	S_{cont}	Recall	Time	$\bar{S}_s$	S_{sn}	Time
IB	0.927	0.065	N2	$0.930, 0.941$	$0.000, 0.059$	0.516	$0.932, 0.941$	$0.762, 0.789$	607
MF	0.915	0.334	FP	$0.856, 0.881$	$0.527, 0.793$	0.734	$0.926, 0.932$	$0.714, 0.751$	579

Source: Andre Paulino de Lima, 2019

the amount of rating data used in the factorisation. Traditionally, the item vectors are extracted from the **Q** matrix obtained by factorising the user-item matrix that includes only the ratings that were allocated to a training partition, and the quality of the item vectors is assessed using ratings allocated to a test partition and according to a performance metric, like recall. We did not adhere to this practice and used the whole of the ratings data to produce item vectors. As this work is focused on the surprise property and is not limited by requirements of commensurability with other system properties, it seemed reasonable to aim for item vectors produced with as much information as possible. This decision was consistently observed in our experiments, so, for example, results for normalised surprise obtained from config C2 is comparable between experiments 3 and 4.

A comparative analysis of the execution times

The time taken to execute the proposed method was about 950 times higher than that of the one plus random. This is in agreement with the relative computational cost estimated in Section 3.5, which predicted the cost to be up to $(2 \times |L_1| + 1)$ times higher because the proposed method needs to compute the minimum and maximum potential surprises (we adopted L_1 with 1,000 items). Despite the large difference, we argue that the method has practical application, since it takes an amount of time that is not incompatible with the task of evaluating a system (5 to 10 minutes on commodity hardware). In addition, it must be noted that the results presented in this experiment, which were obtained from a moderate-sized user sample ($n = 50$ users), are close to those from the previous experiment, in which a much larger sample was employed ($n = 362$).

As a side note, we point out that the value of S_{cont} is near to one, the maximum value it can achieve, whereas the normalised surprise S_{sn} indicates the existence of a larger gap between the current system performance and its maximum capacity.

4.5 Discussion of the evidences

As mentioned in the introduction to this chapter, each of our experiments aimed at one of these objectives: a) to validate a premise of the theoretical model of potential surprise; b) to establish the effect on surprise of item representation and distance function; or c) to compare the proposed and the state-of-the-art method for assessing surprise.

Regarding the first objective, the evidences obtained from the experiments reported on Sections 4.1 (potential surprise of a system) and 4.2.5 (potential surprise of a sequence) suggest that greedy approximations to potential surprise are comparable to their exact counterparts. With respect to the second objective, the evidences obtained from the experiment reported on Section 4.3 showed a large effect on surprise of the factors related to item representation and comparison, as could be anticipated by inspecting the formal definition of surprise in Equation 11. In particular, the analysis of the results revealed some insights about this relationship, such as the link between normalised surprise and the skewness of the pairwise distance distribution employed by the evaluator.

Finally, with regard to the third objective, evidences from the experiment reported on Section 4.4 showed that the method, if applied to estimate system-level, non-normalised surprise, obtains results that are comparable to those in a baseline. In addition, the evidences confirm the difference in computational cost between the state-of-the-art and the proposed method that was anticipated in the theoretical development. However, we argue that, although the difference is large, the method can still be used in practical settings. In addition, in the discussion presented on Section 4.3.4, a low fidelity model was explored to obtain rough estimates of system-level, normalised surprise using item-level, non-normalised surprise, but the conditions of its precision remains as an open question.

5 Conclusion

This project began with a question about over-specialisation in recommender systems: is it possible to measure the degree of over-specialisation of a system? The question naturally led to the notion of serendipity and its components of surprise and relevance, as well as the conception of over-specialisation and "filter bubble" as operating states of the system, where there is expected to be a reduction in the amount of surprise that is embedded in new personalised recommendations. From this new perspective, the initial question was reframed as: is it possible to measure the magnitude of this reduction in surprise? Shortly afterwards, our four research questions took on their current form.

In Chapter 2, several metrics for surprise in recommender systems were reviewed, as well as evaluation methods that can be used in tandem to assess the degree of surprise expressed in either a single item, a sequence of items (i.e., a recommendation list), or a whole system. It was seen that a basic characteristic of these metrics is that surprise involves the notions of item representation and comparison, and these notions are "translated" into an integrated scheme to produce vector representations for items and to assess pairwise (dis)similarity between them. This fact has encouraged a review of basic representation models used in recommender systems, as well as distance functions and recommendation methods that can handle the item vectors produced by the models.

In Chapter 3, the first three research questions were addressed in a deductive way. The chapter starts by revisiting the concept of surprise and draws on a set of properties from the literature on cognitive science that a metric for surprise should have. These properties were used to assist in selecting a metric for surprise in recommender systems. Then, we gradually progressed from the idea of surprise as a finite resource in a system, to the concept of maximum potential surprise to a user as the maximum amount of surprise that a system can offer to an arbitrary user (RQ1), and then to the concept of maximum surprise in a sequence as the maximum amount of surprise that can be embedded in a recommendation list of arbitrary length k (RQ2). These concepts made it possible to create a scale, which was used in the definition of the normalised surprise metric, which reflects the capacity of a system to embed surprise in its recommendations (RQ3).

It is possible to show, from the definition of the normalised surprise metric, that no guarantees can be offered about its invariance to item representation and comparison.

Thus, our fourth research question can be suitably contextualised as follows: how much variation in normalised surprise can be expected if the representation model or the distance function changes (RQ4)? This question was addressed in an empirical way, and the evidence outlined in Chapter 4 shows that there is a large effect on the normalised surprise of item representation and comparison, as would be expected from the definition of the metric. In addition, the evidence shows some interesting patterns, such as the correspondence between negatively skewed pairwise distance distributions and a greater degree of surprise, and the contrasting links between the usual inner product in latent features space and surprise assessed by means of cosine or Euclidean distances in factorisation-based recommenders.

5.1 Limitations and risks

The evaluation method that was proposed in this work basically relies on the item representation model adopted by the system, and can be applied to several categories of recommenders. The representation models that were explored are based on explicit user feedback (i.e., based on ratings) or textual content combined with the items; however, alternative models can be adopted that explore implicit user feedback (e.g., data extracted from traces of user online activity) or hybrid representation approaches (e.g., tag genome proposed by Vig, Sen and Riedl (2012)); the only requirement is that the system must keep item vectors from a single representation space, which is a usual characteristic of recommender systems.

With regard to threats to the validity of this study, we pointed out in Section 2.4 that there is a disconnect between the metrics for surprise in the literature on recommender systems and the characteristics that a metric for surprise should have according to two competing models of surprise described in the literature on cognitive science. In the absence of a consensual metric, we decided to adopt the metric that more closely matches the desired characteristics, and explored several item representations and distance intuitions (including the Kullback-Leibler divergence, as proposed by Itti and Baldi (2009)) to reduce the risks. A controlled environment was employed in the experiments to assure reproducibility of the results, – this consisted of computational simulations. Statistical techniques were employed to assess the results while also seeking to reduce risks to internal validity. In addition, owing to the limited availability of public datasets, the experiments

were based on a single dataset that was extended and enriched to suit our requirements. The experiments examined here should be reproduced with other datasets so that it is possible to assess the external validity of this study.

Another threat to the validity of this study is the adoption of a greedy strategy to obtain approximations of potential surprise. As mentioned in Section 3.3.4, this strategy does not offer theoretical bounds to the quality of its approximations, what may lead to poor estimates of the normalised surprise metrics, $\hat{S}_{sn}$ and $\hat{S}'_{sn}$. In view of this, we tried to mitigate the risk by running two experiments to assess the quality of the obtained estimates: the first was focused on the potential surprise to a user (Experiment 1), and the second was focused on the potential surprise that can be embedded in a sequence (Experiment 2), which is employed by the proposed evaluation method.

5.2 Opportunities for improvement and alternative directions

The proposed evaluation method depends on the ability to obtain an adequate estimate of the potential surprise. It employs a greedy strategy to compute this estimate, at the price of higher computational costs (if compared to the current state-of-the-art evaluation method), and having no theoretical bounds to the accuracy of the approximations. To the best of our knowledge, a problem that is formally similar to the estimation of potential surprise has not been addressed in the literature on optimisation of permutation-based problems. Thus, this provides an opportunity to improve this study by creating an approximation algorithm for potential surprise, with lower computational cost, and preferably with theoretical bounds. A possible starting point is to revise the techniques that were used by the approximation algorithms for the Travelling Salesman Problem (JOHNSON; MCGEOCH, 1995) and a range of longest path problems (REZENDE, 2014).

Another opportunity for extending this work is to explore the link between relevance, surprise, and diversity. As described in Section 2.2, serendipity can be defined as an interaction between surprise and relevance, and it seems reasonable to believe that this is an antagonistic relationship: optimising a recommender for embedding surprise will inevitably reduce expected relevance to unacceptable levels for any practical application, and optimising for relevance alone leads to a reduction in surprise in some situations. However, as far as we know, the effect on relevance of a gradual change in system

parameters towards heightening the degree of surprise is unknown. In principle, an online experiment would be required to show the extent of this effect, which could reveal sweet spots for serendipity. Moreover, it should be noted that the definition of surprise of a sequence (S_s) encompasses the effect of the traditional metric for diversity put forward by Ziegler et al. (2005). According to this definition, diversity is an objective property of a recommendation list, in contrast with the surprise of a sequence, which was defined as a subjective property that results from the interaction between the user and the recommendation. This relationship raises a question about the possibility of increasing diversity without increasing surprise, and vice versa.

To seek consilience with the social sciences, we took some ideas from cognitive science about how human beings experience surprise. As mentioned earlier, there is a gap between how surprise is defined in the literature on recommender systems and the properties that a metric for surprise should have (according to the cognitive models of surprise). This gap could be addressed by the development of models (and metrics) that account for well-known cognitive biases, such as the recall and retrievability biases.

Moreover, there is only a slight correspondence between the experimental settings adopted in user studies in recommender systems and the experimental settings from which evidence is obtained to assess the models of surprise in cognitive science. We argue that the methodological differences raise a question about the validity of self-report studies of user surprise in recommender systems. One alternative line of research is to explore the possible link between surprise and explanatory difficulty to foster the validity. In Foster and Keane (2015), the authors argue that surprise "is a metacognitive estimate of the cognitive work involved in explaining an abnormal event", and present evidences that support the idea that the more surprising the event, the more time the subject takes to produce an explanation. This idea may be combined with an emerging line of research in recommender systems, which seeks to develop methods that can be used to explain the reason a recommendation was produced by means of a textual description (TINTAREV; MASTHOFF, 2015; RANA; BRIDGE, 2018). A starting point would be to conceive an experimental setting that could be used to collect user reaction times (and other measurements) when performing a task of assessing explanations produced by the recommender, and use the data to develop a proxy measure of surprise.

It is worth pointing out that a strictly mathematically exploration of pairwise distance distributions (used in Sections 4.2.4 and 4.3.3 in an intuitive way) may be a

fruitful line of research, since it may make it possible to design an instrument that can show, in qualitative terms, how a distance function can discriminate between items vectors from a finite (metric) space. That may enable the researcher to move from the formal specification of a distance function to obtaining a view of distance as a model that can be suitable for carrying out a particular task.

Finally, there is the opportunity to extend this work by applying the proposed evaluation method to a recommender system that is known to be operating under an over-specialised or filter-bubble condition. The expected result is a time series composed of measurements in which a systematic reduction in surprise is observed. The challenge of pursuing this opportunity is to find a dataset that 1) keeps timestamps for the ratings given by the users, 2) keeps timestamps for the introduction of new items in the repository, and 3) keeps timestamps for the recommendations to which the user has been exposed. The latter requirement is linked to the idea that some users are more prone to follow recommendations than others, and that the variance in behaviour is large (NGUYEN et al., 2014). The Movielens-1M and Movielens-20M datasets meet the first requirement, and the second requirement can be satisfied by taking the movie release date as the date of its first rating, but it does not fulfil the third requirement. In the absence of a dataset that satisfies these characteristics, an alternative method is to create a synthetic one. Moreover, it may be interesting to consider datasets from recommenders that operate in social networks.

5.3 Final remarks

Recommender systems have become ubiquitous technological components in modern society, and their customary role within social networks has extended their influence to a global audience. Their role as massive information filters entails benefits and drawbacks, and as with any technology, the benefits must outweigh the risks. Unfortunately, there are an increasing number of reports ranging from data privacy violations to formally organised political manipulation campaigns and these are extremely detrimental to public life. There must be attempts aimed at reducing the adverse effects that have already taken place and, as is required for resolving any controversial issue, the first step is to reach a sufficiently consensual definition of the phenomenon and the methods needed to assess its scope. We hope this study can contribute to this urgent discussion and its resolution.

Bibliography

ADAMOPOULOS, P.; TUZHILIN, A. On unexpectedness in recommender systems: Or how to expect the unexpected. In: *Proceedings of the Workshop on Novelty and Diversity in Recommender Systems at the Fifth ACM International Conference on Recommender Systems*. New York, NY, USA: ACM, 2011. (DiveRS @ RecSys 2011), p. 11–18. Available at: <https://doi.org/10.1145/2043932.2044019>. Cited 6 times on the pages 22, 37, 39, 42, 44, and 133.

AITCHISON, J. A concise guide to compositional data analysis. Laboratório de Estatística e Geoinformação, 2003. Available at: <http://www.leg.ufpr.br/lib/exe/fetch-.php/pessoais:abtmartins:a_concise_guide_to_compositional_data_analysis.pdf>. Cited 4 times on the pages 48, 49, 52, and 54.

AKIYAMA, T.; OBARA, K.; TANIZAKI, M. Proposal and evaluation of serendipitous recommendation method using general unexpectedness. In: *Proceedings of the Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies at the Fourth ACM International Conference on Recommender Systems*. New York, NY, USA: ACM, 2010. (PRSAT @ RecSys 2010), p. 3–10. Available at: <https://doi-.org/10.1145/1864708.1864795>. Cited 6 times on the pages 22, 37, 39, 40, 42, and 44.

BARONI, M.; DINU, G.; KRUSZEWSKI, G. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014. p. 238–247. Available at: <http://www.aclweb.org/anthology/P14-1023>. Cited 2 times on the pages 32 and 49.

BARTO, A.; MIROLLI, M.; BALDASSARRE, G. Novelty or surprise? *Frontiers in Psychology*, Frontiers, Lausanne, Switzerland, v. 4, p. 907, 2013. ISSN 1664-1078. Available at: <https://www.frontiersin.org/article/10.3389/fpsyg.2013.00907>. Cited on page 22.

BAZERMAN, M. H.; MOORE, D. A. *Judgment in managerial decision making*. 7th. ed. Hoboken, NJ: John Wiley & Sons, 2009. ISBN 978-0470049457. Cited on page 45.

BELIAKOV, G.; CALVO, T.; JAMES, S. Aggregation functions for recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 23, p. 777–808. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_23>. Cited on page 34.

BELLOGIN, A.; CASTELLS, P.; CANTADOR, I. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2011. (RecSys '11), p. 333–336. ISBN 978-1-4503-0683-6. Available at: <http://doi.acm.org/10.1145/2043932-.2043996>. Cited on page 75.

BENGIO, Y. et al. A neural probabilistic language model. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 1137–1155, mar. 2003. ISSN 1532-4435. Available at: <http:-//dl.acm.org/citation.cfm?id=944919.944966>. Cited 2 times on the pages 49 and 50.

BIGI, B. Using kullback-leibler distance for text categorization. In: SEBASTIANI, F. (Ed.). *Advances in Information Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 305–319. ISBN 978-3-540-36618-8. Cited on page 62.

BOUMA, G. Normalized (pointwise) mutual information in collocation extraction. In: *Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology*. Manheim, Germany: GSCL e.V., 2009. (GSCL 2009), p. 31–40. Available at: <https://svn.spraakdata.gu.se/repos/gerlof/pub/www/Docs/npmi-pfd.pdf>. Cited on page 41.

BRADSHAW, S.; HOWARD, P. N. Challenging truth and trust: a global inventory of organized social media manipulation. *The Computational Propaganda Project*, 2018. Cited on page 20.

BRONSHTEIN, I. et al. *Handbook of Mathematics*. 6th. ed. Heilderberg, Germany: Springer Verlag, 2015. ISBN 978-3662462201. Available at: <https://doi.org/10.1007/978-3-662-46221-8>. Cited on page 151.

BURKE, R. Hybrid web recommender systems. In: BRUSILOVSKY, P.; KOBSA, A.; NEJDL, W. (Ed.). *The Adaptive Web: Methods and Strategies of Web Personalization*. Berlin, Heidelberg: Springer, 2007, (Lecture Notes on Computer Science, v. 4321). chap. 12, p. 377–408. ISBN 978-3-540-72079-9. Available at: <https://doi.org/10.1007-/978-3-540-72079-9_12>. Cited 2 times on the pages 21 and 31.

CASTELLS, P.; HURLEY, N. J.; VARGAS, S. Novelty and diversity in recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 26, p. 881–918. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_26>. Cited 2 times on the pages 21 and 36.

CER, D. M. et al. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055, 2017. Available at: <http://arxiv.org/abs/1708.00055>. Cited on page 59.

CREMONESI, P.; KOREN, Y.; TURRIN, R. Performance of recommender algorithms on top-n recommendation tasks. In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2010. (RecSys '10), p. 39–46. ISBN 978-1605589060. Available at: <http://doi.acm.org/10.1145/1864708.1864721>. Cited 5 times on the pages 54, 68, 70, 76, and 101.

CREMONESI, P. et al. An evaluation methodology for collaborative recommender systems. In: *International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*. Washington, DC, USA: IEEE, 2008. (AXMEDIS 2008), p. 224–231. Cited on page 75.

DAI, A. M.; OLAH, C.; LE, Q. V. Document embedding with paragraph vectors. *CoRR*, abs/1507.07998, 2015. Available at: <http://arxiv.org/abs/1507.07998>. Cited on page 51.

DAWSON, M. R. *Mind, body, world: Foundations of cognitive science*. Alberta, Canada: Athabasca University Press, 2013. ISSN 2291-2614. Cited on page 29.

DEZA, M. M.; DEZA, E. *Encyclopedia of Distances*. Heidelberg, Germany: Springer Verlag, 2009. ISBN 978-3642002342. Available at: <https://doi.org/10.1007/978-3-642-00234-2>. Cited 2 times on the pages 41 and 57.

EGOZCUE, J. J. et al. Elements of simplicial linear algebra and geometry. In: PAWLOWSKY-GLAHN, V.; BUCCIANTI, A. (Ed.). *Compositional Data Analysis*. West Sussex, United Kingdom: John Wiley & Sons, 2011. chap. 11, p. 139–157. ISBN 9781119976462. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002-/9781119976462.ch11>. Cited 5 times on the pages 48, 53, 62, 63, and 64.

ELLIS, P. D. *The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results*. Cambridge, UK: Cambridge University Press, 2010. ISBN 978-0521142465. Cited on page 121.

FELFERNIG, A. et al. Constraint-based recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 5, p. 161–190. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_5>. Cited 2 times on the pages 33 and 34.

FERRO, N. et al. From Evaluating to Forecasting Performance: How to Turn Information Retrieval, Natural Language Processing and Recommender Systems into Predictive Sciences (Dagstuhl Perspectives Workshop 17442). *Dagstuhl Manifestos*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, v. 7, n. 1, p. 96–139, 2018. ISSN 2193-2433. Available at: <http://drops.dagstuhl.de/opus/volltexte/2018/9898>. Cited on page 29.

FOSTER, M. I.; KEANE, M. T. Why some surprises are more surprising than others: Surprise as a metacognitive sense of explanatory difficulty. *Cognitive Psychology*, v. 81, p. 74 – 116, 2015. ISSN 0010-0285. Available at: <http://www.sciencedirect-.com/science/article/pii/S0010028515000626>. Cited 3 times on the pages 43, 86, and 141.

GE, M.; DELGADO-BATTENFELD, C.; JANNACH, D. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2010. (RecSys '10), p. 257–260. ISBN 978-1-60558-906-0. Available at: <http://doi.acm-.org/10.1145/1864708.1864761>. Cited 6 times on the pages 22, 37, 38, 42, 44, and 133.

GEMMIS, M. de et al. Semantics-aware content-based recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 4, p. 119–160. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_4>. Cited 2 times on the pages 32 and 39.

GUNAWARDANA, A.; SHANI, G. Evaluating recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 8, p. 265–308. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_8>. Cited on page 73.

GUP, T. Technology and the end of serendipity. *The Chronicle of Higher Education*, v. 44, n. 21, p. A52, 1997. Cited on page 20.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, ACM, New York, NY, USA, v. 5, n. 4, p. 19:1–19:19, dez. 2015. ISSN 2160-6455. Available at: <http://doi.acm.org/10.1145/2827872>. Cited 2 times on the pages 27 and 103.

HERLOCKER, J.; KONSTAN, J. A.; RIEDL, J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, v. 5, n. 4, p. 287–310, Oct 2002. ISSN 1573-7659. Available at: <https://doi.org/10.1023-/A:1020443909834>. Cited 2 times on the pages 22 and 35.

HERLOCKER, J. L. et al. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, ACM, New York, NY, USA, v. 22, n. 1, p. 5–53, jan. 2004. ISSN 1046-8188. Available at: <http://doi.acm.org/10.1145/963770.963772>. Cited on page 22.

HILL, F.; CHO, K.; KORHONEN, A. Learning distributed representations of sentences from unlabelled data. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016. p. 1367–1377. Available at: <http://www.aclweb.org/anthology/N16-1162>. Cited on page 52.

ITTI, L.; BALDI, P. Bayesian surprise attracts human attention. *Vision Research*, v. 49, n. 10, p. 1295 – 1306, 2009. ISSN 0042-6989. Visual Attention: Psychophysics, electrophysiology and neuroimaging. Available at: <http://www.sciencedirect.com-/science/article/pii/S0042698908004380>. Cited 6 times on the pages 22, 43, 44, 56, 86, and 139.

JACCARD, P. The distribution of the flora in the alpine zone. *New phytologist*, Wiley Online Library, v. 11, n. 2, p. 37–50, 1912. Cited on page 60.

JAWAHAR, G. Improving distributed representations of tweets - present and future. In: *Proceedings of ACL 2017, Student Research Workshop*. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 4–10. Available at: <http://aclweb.org/anthology/P17-3002>. Cited 2 times on the pages 51 and 52.

JOHN, O. P.; BENET-MARTINEZ, V. Measurement: Reliability, construct validation, and scale construction. In: REIS, H. T.; JUDD, C. M. (Ed.). *Handbook of Research Methods in Social and Personality Psychology*. Cambridge, MA: Cambridge University Press, 2000. chap. 13, p. 339–369. ISBN 0521559030. Cited on page 36.

JOHNSON, D. S.; MCGEOCH, L. A. The traveling salesman problem: A case study in local optimisation (preliminary book version). Chichester, UK, 1995. Available at: <http://www.csc.kth.se/utbildning/kth/kurser/DD2440/avalg14/TSP-JohMcg97.pdf>. Cited on page 140.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing (draft manuscript to the 3rd. edition)*. 2018. Available at: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>. Cited 2 times on the pages 59 and 60.

KAMINSKAS, M.; BRIDGE, D. Measuring surprise in recommender systems. In: *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design, at the 8th ACM Conference on Recommender Systems*. New York, NY, USA:

ACM, 2014. (REDD @ RecSys '14), p. 393–394. ISBN 978-1-4503-2668-1. Available at: <http://doi.acm.org/10.1145/2645710.2645780>. Cited 14 times on the pages 22, 28, 37, 40, 42, 44, 60, 78, 81, 83, 96, 97, 101, and 133.

KAMINSKAS, M.; BRIDGE, D. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, ACM, New York, NY, USA, v. 7, n. 1, p. 2:1–2:42, dez. 2016. ISSN 2160-6455. Available at: <http://doi.acm.org/10.1145/2926720>. Cited 2 times on the pages 22 and 36.

KOREN, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2008. (KDD '08), p. 426–434. ISBN 978-1-60558-193-4. Available at: <http://doi.acm.org/10.1145/1401890.1401944>. Cited on page 75.

KOREN, Y.; BELL, R. Advances in collaborative filtering. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 3, p. 77–118. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_3>. Cited 4 times on the pages 32, 53, 54, and 68.

LEE, L. Measures of distributional similarity. In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1999. (ACL '99), p. 25–32. ISBN 1-55860-609-3. Available at: <https://doi.org/10.3115/1034678.1034693>. Cited on page 60.

LEVY, O.; GOLDBERG, Y. Neural word embedding as implicit matrix factorization. In: GHAHRAMANI, Z. et al. (Ed.). *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014. p. 2177–2185. Available at: <http://papers.nips.cc/paper-/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf>. Cited 2 times on the pages 52 and 108.

LORENZI, F.; RICCI, F. Case-based recommender systems: A unifying view. In: *Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization*. Berlin, Heidelberg: Springer-Verlag, 2005. (ITWP'03), p. 89–113. ISBN 3-540-29846-0, 978-3-540-29846-5. Available at: <https://doi.org/10.1007/11577935_5>. Cited on page 34.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. New York, NY: Cambridge University Press, 2008. ISBN 978-0521865715. Cited 3 times on the pages 47, 49, and 59.

MARTÍN-FERNÁNDEZ, J. A.; PALAREA-ALBALADEJO, J.; OLEA, R. A. Dealing with zeros. In: PAWLOWSKY-GLAHN, V.; BUCCIANTI, A. (Ed.). *Compositional Data Analysis*. West Sussex, United Kingdom: John Wiley & Sons, 2011. chap. 4, p. 43–58. ISBN 9781119976462. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002-/9781119976462.ch4>. Cited on page 62.

MCNEE, S. M.; RIEDL, J.; KONSTAN, J. A. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In: *Extended Abstracts on Human Factors in*

Computing Systems. New York, NY, USA: ACM, 2006. (CHI EA '06), p. 1097–1101. ISBN 1-59593-298-4. Available at: <http://doi.acm.org/10.1145/1125451.1125659>. Cited on page 22.

MEYER, W.-U.; REISENZEIN, R.; SCHÜTZWOHL, A. Toward a process analysis of emotions: The case of surprise. *Motivation and Emotion*, v. 21, n. 3, p. 251–274, Sep 1997. ISSN 1573-6644. Available at: <https://doi.org/10.1023/A:1024422330338>. Cited on page 43.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. Available at: <http://arxiv.org/abs/1301.3781>. Cited on page 50.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: BURGES, C. J. C. et al. (Ed.). *Advances in Neural Information Processing Systems 26*. Red Hook, NY, USA: Curran Associates, Inc., 2013. p. 3111–3119. Cited 3 times on the pages 51, 108, and 156.

MONTGOMERY, D. C. *Design and analysis of experiments*. 8th. ed. Singapore: John Wiley & Sons, 2017. ISBN 978-8126540501. Cited on page 116.

MURAKAMI, T.; MORI, K.; ORIHARA, R. Metrics for evaluating the serendipity of recommendation lists. In: SATOH, K. et al. (Ed.). *New Frontiers in Artificial Intelligence*. Berlin, Heidelberg: Springer, 2008. p. 40–46. ISBN 978-3-540-78197-4. Available at: <https://doi.org/10.1007/978-3-540-78197-4_5>. Cited 6 times on the pages 22, 37, 38, 41, 44, and 133.

NAKATSUJI, M. et al. Classical music for rock fans?: Novel recommendations for expanding user interests. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2010. (CIKM '10), p. 949–958. ISBN 978-1-4503-0099-5. Available at: <http://doi.acm.org/10.1145/1871437-.1871558>. Cited on page 36.

NGUYEN, T. T. et al. Exploring the filter bubble: The effect of using recommender systems on content diversity. In: *Proceedings of the 23rd International Conference on World Wide Web*. New York, NY, USA: ACM, 2014. (WWW '14), p. 677–686. ISBN 978-1-4503-2744-2. Available at: <http://doi.acm.org/10.1145/2566486.2568012>. Cited 5 times on the pages 20, 32, 94, 95, and 142.

NING, X.; DESROSIERS, C.; KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 2, p. 37–76. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_2>. Cited 2 times on the pages 32 and 53.

O'SEARCOID, M. *Metric Spaces*. London, UK: Springer Verlag, 2007. ISBN 9781846286278. Available at: <https://doi.org/10.1007/978-1-84628-627-8>. Cited on page 57.

PARISER, E. *The filter bubble: What the Internet is hiding from you*. New York, NY: Penguin US, 2011. ISBN 978-1452651811. Cited on page 20.

PAZZANI, M. J. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, v. 13, n. 5, p. 393–408, Dec 1999. ISSN 1573-7462. Available at: <https://doi.org/10.1023/A:1006544522159>. Cited on page 34.

RANA, A.; BRIDGE, D. Explanations that are intrinsic to recommendations. In: *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. New York, NY, USA: ACM, 2018. (UMAP '18), p. 187–195. ISBN 978-1-4503-5589-6. Available at: <http://doi.acm.org/10.1145/3209219.3209230>. Cited on page 141.

ŘEHŮŘEK, R.; SOJKA, P. Software Framework for Topic Modelling with Large Corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, 2010. p. 45–50. `http://is.muni.cz/publication/884893/en`. Cited 2 times on the pages 108 and 156.

REISENZEIN, R.; HORSTMANN, G.; SCHÜTZWOHL, A. The cognitive-evolutionary model of surprise: A review of the evidence. *Topics in Cognitive Science*, 2017. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12292>. Cited 4 times on the pages 22, 43, 45, and 86.

REZENDE, S. F. de. *Longest path in graphs*. Dissertação (Mestrado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil, 2014. Cited on page 140.

SILVEIRA, T. et al. A framework for unexpectedness evaluation in recommendation. In: *Proceedings of the Symposium on Applied Computing*. New York, NY, USA: ACM, 2017. (SAC '17), p. 1662–1667. ISBN 978-1-4503-4486-9. Available at: <http://doi.acm.org/10.1145/3019612.3019760>. Cited 2 times on the pages 38 and 41.

TINTAREV, N.; MASTHOFF, J. Explaining recommendations: Design and evaluation. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. 2nd. ed. Boston, MA: Springer US, 2015. chap. 10, p. 353–382. ISBN 978-1-4899-7637-6. Available at: <https://doi.org/10.1007/978-1-4899-7637-6_10>. Cited on page 141.

TURNEY, P. D.; PANTEL, P. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, AI Access Foundation, USA, v. 37, n. 1, p. 141–188, jan 2010. ISSN 1076-9757. Available at: <http://dl.acm-.org/citation.cfm?id = 1861751.1861756>. Cited 4 times on the pages 32, 47, 48, and 49.

TVERSKY, A. Features of similarity. *Psychological Review*, v. 84, n. 4, p. 327–352, 1977. Available at: <http://dx.doi.org/10.1037/0033-295X.84.4.327>. Cited on page 45.

TVERSKY, A.; KAHNEMAN, D. Judgment under uncertainty: Heuristics and biases. *Science*, American Association for the Advancement of Science, v. 185, n. 4157, p. 1124–1131, 1974. ISSN 0036-8075. Available at: <http://science.sciencemag.org/content-/185/4157/1124>. Cited on page 45.

VARGAS, S.; CASTELLS, P. Rank and relevance in novelty and diversity metrics for recommender systems. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2011. (RecSys '11), p. 109–116. ISBN 978-1-4503-0683-6. Available at: <http://doi.acm.org/10.1145/2043932.2043955>. Cited on page 36.

VELLEMAN, P. F.; WILKINSON, L. Nominal, ordinal, interval, and ratio typologies are misleading. *The American Statistician*, Taylor & Francis, v. 47, n. 1, p. 65–72, 1993. Available at: <https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1993-.10475938>. Cited on page 46.

VIG, J.; SEN, S.; RIEDL, J. The tag genome: Encoding community knowledge to support novel interaction. *ACM Trans. Interact. Intell. Syst.*, ACM, New York, NY, USA, v. 2, n. 3, p. 13:1–13:44, set. 2012. ISSN 2160-6455. Available at: <http://doi.acm.org/10.1145/2362394.2362395>. Cited on page 139.

ZHANG, Y. C. et al. Auralist: Introducing serendipity into music recommendation. In: *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2012. (WSDM '12), p. 13–22. ISBN 978-1-4503-0747-5. Available at: <http://doi.acm.org/10.1145/2124295.2124300>. Cited 5 times on the pages 22, 37, 40, 42, and 44.

ZIEGLER, C.-N. et al. Improving recommendation lists through topic diversification. In: *Proceedings of the 14th International Conference on World Wide Web*. New York, NY, USA: ACM, 2005. (WWW '05), p. 22–32. ISBN 1-59593-046-9. Available at: <http://doi.acm.org/10.1145/1060745.1060754>. Cited on page 141.

APPENDIX A – Formal definition of the proposed metrics

On the basis of the conditions described in Scenario 3.3, which were assumed by the potential surprise model, let I be a finite set of items (Condition 1), and let u be a user that has been exposed to at least one item (Condition 3). Then, the set of items known to the user u, namely E_u, is an element in the power set of I, except for the empty set:

$$E_u \in \mathscr{P}^*(I), \text{ with } \mathscr{P}^*(I) = \mathscr{P}(I) \setminus \varnothing,$$

and the set of items that are unknown to the user u, namely N_u, is the complement of E_u:

$$N_u = I \setminus E_u = \overline{E_u}.$$

Let v be a function that maps each item in I to its vector representation:

$$\begin{aligned} v\colon I &\to \mathbb{R}^n \\ i &\mapsto \mathbf{M}_i, \end{aligned}$$

where $\mathbf{M}_i$ is the i^{th} row (or column) of a matrix $\mathbf{M}$, which was created using one of the representation models that were described in Section 2.5. Assume that the representation model was properly tuned so that different item vectors were produced for distinct items:

$$i \neq j \iff \mathbf{M}_i \neq \mathbf{M}_j.$$

The surprise of an item, S_i, is a function that computes the minimum distance between the vector representations respective to an item i and the items in E_u:

$$\begin{aligned} S_i\colon I \times \mathscr{P}^*(I) &\to \mathbb{R}_+ \\ (i, E_u) &\mapsto \min_{j \in E_u} dist(v(i), v(j)), \end{aligned}$$

where *dist* is a distance function that is compatible to the choice of $\mathbf{M}$, as described in Section 2.6. It must be noted that if we allowed E_u to be mapped to an empty set, this function would not be defined, and this justifies the Condition 3 of the theoretical model.

We now focus on defining a sequence of items. Let A be a non-empty subset of I. Then $A \in \mathscr{P}^*(I)$. The set of all the sequences of length k composed of items in A, referred to as arrangements of A with length k (BRONSHTEIN et al., 2015), is defined as:

$$\mathscr{A}_k(A) = \bigcup_{A' \in C_k^A} \Pi(A'),$$

where C_k^A is the set of all subsets of A that contains k elements of A (combinations), and $\Pi(A')$ is the set containing all permutations of the elements in the set A'. For $k = 0$, assume

that $\mathscr{A}_k(A) = \{\epsilon\}$, where ϵ is an empty sequence. For $k > |A|$, assume that $\mathscr{A}_k(A) = \varnothing$.

Example. Suppose $A = \{a, b, c\}$ and $k = 2$. Then $C_k^A = \{\{a, b\}, \{a, c\}, \{b, c\}\}$, $\Pi(\{a, b\}) = \{(a, b), (b, a)\}$, $\Pi(\{a, c\}) = \{(a, c), (c, a)\}$, $\Pi(\{b, c\}) = \{(b, c), (c, b)\}$, and $A_k(A) = \{(a, b), (b, a), (a, c), (c, a), (b, c), (c, b)\}$.

We extend on the previous definition to include the set of all sequences of arbitrary length composed of items in A, referred to as the arrangements of A, and defined as:

$$\mathscr{A}(A) = \bigcup_{k=0}^{|A|} \mathscr{A}_k(A).$$

Example. Suppose $A = \{a, b, c\}$. Then:

- for $k = 0$, $\mathscr{A}_k(A) = \{\epsilon\}$.
- for $k = 1$, $\mathscr{A}_k(A) = \{(a), (b), (c)\}$.
- for $k = 2$, $\mathscr{A}_k(A) = \{(a, b), (b, a), (a, c), (c, a), (b, c), (c, b)\}$.
- for $k = 3$, $\mathscr{A}_k(A) = \{(a, b, c), (a, c, b), (c, a, b), (c, b, a), (b, c, a), (b, a, c)\}$.

Thus, $\mathscr{A}(A) = \{\epsilon, (a), (b), (c), (a, b), (b, a), (a, c), (c, a), (b, c), (c, b), (a, b, c), (a, c, b), (c, a, b), (c, b, a), (b, c, a), (b, a, c)\}$.

Note that the predicate $\mathscr{A}$ can be used to specify all the sequences a recommender system can generate to an arbitrary user u that, in agreement with the Condition 5, must be composed of items unknown to them: $\mathscr{A}(\overline{E_u})$. We can build on this idea and define a predicate to identify all the sequences a recommender can generate to each of its users:

$$\mathscr{S}(I) = \{(seq, E_u) : E_u \in \mathscr{P}^*(I), seq \in \mathscr{A}(\overline{E_u})\},$$

and we refer to the predicate $\mathscr{S}(I)$ as the recommendation space of I. Each element in the recommendation space combines a specific user past experience (E_u) with a potential recommendation list seq that the recommender system can produce for them.

Using the recommendation space, we can define the surprise of a sequence as:

$$\begin{aligned} &S_s \colon \mathscr{S}(I) \to \mathbb{R}_+ \\ &(seq, E_u) \mapsto \begin{cases} S_i(h, E_u) + S_s(t, E_u \cup \{h\}), & \text{if } |seq| > 0, \\ 0, & \text{if } |seq| = 0, \end{cases} \end{aligned}$$

where h represents the head of the sequence seq, namely seq_1, and t its remaining items, $(seq_2, \ldots, seq_m)$. This formulation was presented in Equation 35 (Section 3.3.1).

Analogously, we can define the maximum potential surprise of a recommendation list of length k produced for an arbitrary user u as:

$$S'_{pmax}\colon \mathscr{P}^*(I) \times \{0, \ldots, |I|-1\} \to \mathbb{R}_+$$
$$(E_u, k) \mapsto \begin{cases} \max\limits_{seq \in \mathscr{A}_k(\overline{E_u})} S_s(seq, E_u), & \text{if } k \leq |\overline{E_u}|, \\ 0, & \text{if } k > |\overline{E_u}|. \end{cases}$$

In this definition, $k \in \{0, \ldots, |I|-1\}$ because k represents the length of a sequence composed of items unknown to the user. Then, $k \leq |\overline{E_u}| \implies 0 \leq k \leq |I|-1$. Note that if $k = |\overline{E_u}|$, then this function calculates the maximum potential surprise the recommender system can offer to the user u, and it becomes equivalent to the predicate S_{pmax} presented in Equation 36 (Section 3.3.2).

Similarly, we can define the minimum potential surprise of a recommendation list produced for an arbitrary user u as:

$$S'_{pmin}\colon \mathscr{P}^*(I) \times \{0, \ldots, |I|-1\} \to \mathbb{R}_+$$
$$(E_u, k) \mapsto \begin{cases} \min\limits_{seq \in \mathscr{A}_k(\overline{E_u})} S_s(seq, E_u), & \text{if } k \leq |\overline{E_u}|, \\ 0, & \text{if } k > |\overline{E_u}|. \end{cases}$$

If $k = |\overline{E_u}|$, then this function computes the minimum potential surprise the recommender system can offer to the user u, which corresponds to the value calculated by the predicate S_{pmin} presented in Equation 37 (Section 3.3.2).

Finally, we can define the normalised potential surprise of a recommendation list produced for an arbitrary user u as:

$$S'_{sn}\colon \mathscr{S}(I) \to [0,1] \subset \mathbb{R}_+$$
$$(seq, E_u) \mapsto \frac{S_s(seq, E_u) \quad - S'_{pmin}(E_u, |seq|)}{S'_{pmax}(E_u, |seq|) - S'_{pmin}(E_u, |seq|)}.$$

If $k = |\overline{E_u}|$, then this function computes the normalised potential surprise of a recommendation list containing all items unknown to the user u (total sequence), which corresponds to the value calculated by the predicate S_{sn} presented in Equation 38 (Section 3.3.3). It must be noted that the predicate S'_{sn} is not defined when its denominator is zero. In this situation, it holds that:

$$S'_{pmin}(E_u, k) = S'_{pmax}(E_u, k) \iff S_s(seq, E_u) = v, \forall\, seq \in \mathscr{A}_k(\overline{E_u}),$$

which means that every sequence of length $k = |seq|$ in $\mathscr{A}_k(\overline{E_u})$ embeds the same amount of surprise, and v corresponds to the maximum and the minimum values of potential surprise. In case of a singularity, we assume that $S'_{sn}(seq, E_u) = 1$ by convention.

■

APPENDIX B – Sample size for a population mean

The controlled environment coordinates the application of evaluation methods to a single recommender system instance. To avoid the computational cost of having to consider all users in the evaluation, the methods employ a small random sample of users, which is drawn by the controlled environment. In our case, the target estimate of the methods is the mean value of a property in the population, reported as a confidence interval:

$$\alpha \approx Pr(-z \leq \frac{\sqrt{n}}{s}(\overline{X} - \mu) \leq +z) \implies \alpha \approx Pr\left(\mu \in \left[\overline{X} - \frac{zs}{\sqrt{n}}, \overline{X} + \frac{zs}{\sqrt{n}}\right]\right), \text{ where:}$$

- The variable α reflects the aimed confidence level. For the traditional confidence level of $\alpha = 0.95$, and large samples ($n > 30$), the value of z is taken as 1.96;
- The variable μ is the true population mean, which is the parameter being estimated by $\overline{X}$, the sample mean;
- The variable s is an estimator to the true standard deviation of the population, σ, which is generally obtained from a sample of the population.

The size of the sample, n, which must be large enough to avoid an unacceptable level of confidence in the estimate, as well as an inadequate precision, is computed as:

$$n = \left(\frac{zs}{E}\right)^2, \tag{51}$$

where the variable E reflects the aimed margin of error, which is expressed in the same measurement unit of the estimated standard deviation s.

It must be noted that the estimate for the standard deviation varies substantially across experimental conditions: from $s = 0.039$ in C0FP to $s = 0.136$ in U5N3. According to the principle that sources of variation in an experiment must be controlled when possible, one must decide if obtaining estimates from same-sized samples (i.e., n is constant across conditions) is more important than obtaining estimates with same-length intervals (i.e., $zs/\sqrt{n}$ is constant across conditions). Ours is the first case since the experiment follows a within-subjects design. In this design, each test user is allocated to all conditions. To ensure this requirement is satisfied, a single random sample of n users is drawn once by the controlled environment and reused across conditions. As a result, the margin of error will vary across conditions. To solve the conflict between the requirement for keeping n constant and the reality of having varying estimates for s, we defined an upper bound for precision by resorting to an overestimated upper bound for s ($s^* = 0.5, E^* = 0.05$). Finally, since the population size, ps, is known and finite, we adjust the sample size estimator to:

$$n' = \left\lceil \frac{n}{1 + (n-1)/ps} \right\rceil. \tag{52}$$

Application. There are 6,014 eligible users in our extended dataset. Assume $s^* = 0.5$ as an upper bound to the estimated standard deviation (s) obtained from any experimental condition. According to Equations 51 and 52, to obtain 95% confidence level estimate for $\hat{S}'_{sn}$ with a margin of error of $E^* = 0.05$, we need a sample containing:

$$n = \left(\frac{zs^*}{E^*}\right)^2 = \left(\frac{1.96 \times 0.5}{0.05}\right)^2 \approx 384 \text{ test users,}$$
$$n' = \left\lceil \frac{384}{1 + 383/6,014} \right\rceil = \left\lceil 361.2 \right\rceil = 362 \text{ test users.}$$

Considering a random sample containing $n = 362$ measurements of $\hat{S}'_{sn}$ obtained from the config C0FP ($\overline{X} = 0.306, s = 0.039$), the 95% confidence level estimate corresponds to:

$$0.95 \approx Pr\left(\mu \in \left[0.306 - \frac{1.96 \times 0.039}{\sqrt{362}}, 0.306 + \frac{1.96 \times 0.039}{\sqrt{362}}\right]\right) = Pr(\mu \in [0.302, 0.310]).$$

The actual margin of error is $E = zs/\sqrt{n} = (1.96 \times 0.039)/\sqrt{362} = 0.004 < 0.05 = E^*$.

Considering a random sample containing $n = 362$ measurements of $\hat{S}'_{sn}$ obtained from the config U5N3 ($\overline{X} = 0.432, s = 0.136$), the 95% confidence level estimate corresponds to:

$$0.95 \approx Pr\left(\mu \in \left[0.432 - \frac{1.96 \times 0.136}{\sqrt{362}}, 0.432 + \frac{1.96 \times 0.136}{\sqrt{362}}\right]\right) = Pr(\mu \in [0.418, 0.446]).$$

The actual margin of error is $E = zs/\sqrt{n} = (1.96 \times 0.136)/\sqrt{362} = 0.014 < 0.05 = E^*$.

■

APPENDIX C – Doc2Vec parameters employed in Model D

In Section 4.2.2, an item representation model which is based on the document-term matrix was described (Model D). This representation model produces lower-dimensional item vectors from the document-term matrix by applying an implementation of the `Doc2Vec` algorithm (MIKOLOV et al., 2013b) that has generously been made available by the Gensim company (ŘEHŮŘEK; SOJKA, 2010). This implementation has many parameters, and in the following we describe the parameters that were employed to produce the Model D:

Package: gensim 2.3.0 on Python 3.6.2 with numpy 1.13 (`np113py36_0`)

- Parameter `dm` (set to 0): uses the distributed bag of words algorithm (PV-DBOW).
- Parameter `size` (set to 100): produces 100-dimensional item vectors.
- Parameter `window` (set to 5): length of the word context ($|w_{t-n+1}^{t-1}| = 6$).
- Parameter `alpha` (set to 0.025): initial learning rate.
- Parameter `min_alpha` (set to 0.001): minimum learning rate.
- Parameter `min_count` (set to 2): lower bound to word frequency.
- Parameter `sample` (set to 0.0): no downsampling for high frequency words.
- Parameter `max_vocab_size` (set to None): vocabulary size can grow as needed.
- Parameter `hs` (set to zero): uses negative sampling.
- Parameter `negative` (set to 5): number of "noise" words employed in the training.
- Parameter `dm_mean` (set to 0): uses the sum of the context vectors.
- Parameter `dm_concat` (set to 0): uses non-concatenative mode.
- Parameter `dm_tag_count` (set to 1): each item (document) has a unique tag.
- Parameter `dbow_words` (set to 0): does not train word vectors simultaneously.
- Parameter `iter` (set to 100): number of training epochs.
- Parameter `seed` (set to 23): seed of the internal random number generator.

Any parameters not listed above were kept at their default values. It must be noted that in the current gensim package version (3.3.0), this API has been made available as the `models.deprecated.doc2vec` class, but new users are advised to adopt `models.doc2vec` class instead, which has a slightly different API.

■