# Generating personalized data narrations from EDA notebooks

Alexandre Chanson, Faten El Outa, Nicolas Labroche, Patrick Marcel, Verónika Peralta, Willeme Verdeaux
University of Tours
Blois, France
firstName.lastName@univ-tours.fr

Lucile Jacquemart
University of Tours
Blois, France
Lucile.Jacquemart@etu.univ-tours.fr

## ABSTRACT

In this short paper, we present our preliminary results for generating personalized data narrations by extracting messages from a collection of Exploratory Data Analysis (EDA) notebooks over a given dataset. The approach consists of extracting features from notebooks to learn what interesting messages they expose. Based on those interesting messages, we formalize the problem of producing a user-tailored data narration, i.e., a coherent sequence of messages matching a given user profile. We developed a proof of concept and experimented with Kaggle.com notebooks.

## 1 INTRODUCTION

Exploratory Data Analysis (EDA) is the notoriously tedious task of interactively analyzing datasets to gain insights [10]. EDA notebooks are shared curated, illustrative EDA sessions prepared by data scientists [6, 17]. EDA notebooks are essentially sequences of programmatic operations and their commented results, shared on code sharing platforms such as Kaggle[1]. Supporting EDA can be done by pre-analyzing datasets for computing insights [20] or by automatically generating EDA notebooks using deep learning [6].

Data narration is the activity of producing narratives supported by facts extracted from data exploration and analysis, using interactive visualizations [1]. In an effort to clarify the concepts of data narratives, we recently defined a data narrative *as a structured composition of messages that (a) convey findings over the data, and, (b) are typically delivered via visual means in order to facilitate their reception by an intended audience*, and we proposed a conceptual model describing and structuring the key concepts around data narratives [15]. While several works informally describe the process of data narration crafting [3, 11], automated data narration only starts to gain attention [8, 19].

Our present work contributes to the field of automated data narration, and aims at connecting EDA notebooks to data narrations. More precisely, our objective is to construct data narrations from EDA notebooks. This requires to (i) identify messages that convey findings in the data, (ii) ensure they are relevant for a given user profile, (iii) arrange them in a coherent composition, and (iv) present them visually.

Problem (i) is addressed by formally defining a message as a component of an EDA notebook, extracting them and learning a model of message interestingness. Problem (ii) is addressed by representing messages and user profiles in a vector space, using a classical TF-IDF representation, and using Cosine similarity to select messages closest to the user profile. Problem (iii) is formalized as an instance of the Traveling Analyst Problem (TAP) [2]. Finally, Visual presentation (iv) is ensured by reusing existing visualizations extracted from notebooks.

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

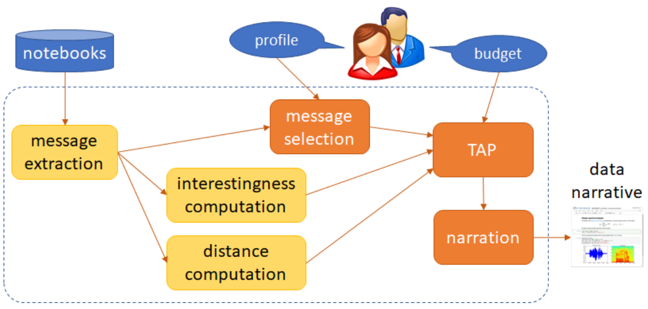

**Figure 1: Overview of the approach**

The general pipeline of our approach is shown in Figure 1.

There are three offline computation modules that deal, respectively, with message extraction, computation of message interestingness, and computation of the cognitive distance between messages. These computations are useful for ensuring that messages in the data narrative are interesting and are structured in a cognitively-coherent way. Then, online, for a given user need, the message selection module preselects messages that match the user's profile. The user also specifies a budget, representing the maximum number of messages to be included in the data narrative. The TAP module takes as input the preselected messages, their interestingness and distance scores, and the budget, and produces an ordered list of messages (taken among the preselected ones) that maximize the overall interestingness and minimize their cognitive distance, while satisfying the given budget. Finally, the narration module generates the data narrative.

Our contributions include:

- a formal framework,
- learning the interestingness of messages,
- an algorithm to generate a user-tailored data narrative from a set of notebooks,
- a proof of concept with Kaggle notebooks, producing various data narratives for a given dataset.

The paper is organized as follows. The next section reviews related works. Section 3 provides the formal background and describes the features we consider to learn messages' interestingness. Section 4 formalizes the problem and presents our solution. Section 5 discusses the implementation and tests. Finally, Section 6 concludes and draw perspectives.

## 2 RELATED WORK

In this section we review related work pertaining to the generation of data narratives or the automation of part of the process.

## 2.1 Automating data narration

Firstly, several works propose solutions for automating data narration starting from a user query [8], a spreadsheet [19], or a topic [18].

The precursor work of Gkesoulis et al. [8] introduced CineCubes, a system that allows the automatic generation of a data story over an OLAP database, with a simple user query as starting point. Each data story has three acts. The first providing contextualization for the characters as well as the incident that sets the story on the move, the second where the protagonists and the rest of the roles build up their actions and reactions and the third where the resolution of the film is taking place. The first one refers to the execution of the original query provided by the user. The second act exploits the selection conditions of the original query and automatically generates comparative drill-up queries to provide contextualization and finally, the third act drills down in the grouping levels of the original result to see the breakdown of its (aggregate) measures and understand its internal structure to provide further analysis of the results. Their tests revealed the ability of Cinecubes to generate a fast report of better quality. However, its fixed structure in three acts can only produce simple data stories with limited insights and visualizations.

Shi et al. [19] proposed Calliope, a system that automatically generates visual data stories from an input spreadsheet. The system incorporates a new logic-oriented Monte Carlo tree search algorithm that explores the data space given by the input spreadsheet to progressively generate story pieces (i.e., data facts) and organize them in a logical sequence. The importance of data facts is measured based on information theory. Each data fact is visualized in a chart and captioned by an automatically generated description. A user study highlighted that the logical order is consistent to humans, the generated data story express useful data insights, and the visualization modes are satisfactory. Nevertheless, Calliope cannot understand data semantics to better generate the story contents and logic. Also, the generated captions are too rigid and contain grammar errors, and the visual encoding generated are notably simple.

Shi et al. [18] proposed AutoClips, an automatic approach to generate data videos from a given topic. It is based on 4 phases: (i) *collecting a series of data facts* around a certain topic, (ii) *constructing a storyline* as an assembly of these data facts into a sequence, (iii) *choosing data visualizations* for the data facts and deciding how to animate them by drawing a storyboard, and finally, (iv) *realizing the storyboard* via a design software in which the narrator edits and combines the animated visualizations until a coherent data video is accomplished. Their evaluation revealed that AutoClips can generate comprehensible and engaging data videos which have comparable quality with human-made videos. However, the system only supports tabular data and favors datasets with diverse column types.

Wang et al. [22] conducted a qualitative analysis on 245 infographics studying the design space in terms of structures, sheet layouts, fact types, and visualization styles. Based on those, the authors propose a system for the auto-generation of fact sheet generation. It consists of three phases: (i) fact extraction, (ii) fact composition, and (iii) presentation synthesis. Their validation of the system highlighted the efficiency of data exploration and the ease of understanding of the visualizations. As limitations, we point out that data semantics is not considered during exploration, and that visualizations are taken from a small-sized predefined library.

## 2.2 Automatic data exploration

Some works [6, 12] propose solutions for automating data exploration, the first step of data narration.

McAuley et al. [12] propose ExploroBOT, a novel system developed to support rapid exploration using a combination of automatic chart generation and intuitive navigation supported by a novel visual guidance framework. The criteria to quantify the interestingness of chart are: (i) data correlation: highly correlated data in scatter plots and trend charts, hints towards an interesting relationship between the two variables. (ii) Peaks: Spikes and large differences in a numerical attributes instantly attract attention. (iii) Outliers: A chart with more outliers is deemed more interesting.

El et al. [6] proposed ATENA, a system that takes an input dataset and auto-generates a compelling exploratory session, presented in an EDA notebook. They shaped EDA into a control problem, and devised a novel Deep Reinforcement Learning architecture to effectively optimize the notebook generation.

Personnaz et al. [16] introduce DORA the explorer, which provides guidance to data explorer relying on Deep Reinforcement Learning that combines intrinsic (curiosity) and extrinsic (familiarity) rewards.

Finally, Deutch et al. [5] deal with the generation of explanations for highlighting exploration results. They proposed ExplainED, a system for automatically explaining views in EDA notebooks. The explanations are presented in Natural Language and describe the particular elements of the view that are the most interesting (the ones having the highest Shapley values).

To the best of our knowledge, our work is the first aiming at automating the production of personalized data narrative by leveraging existing EDA notebooks. One prominent aspect of our approach is to qualify the interestingness of messages contained in existing notebooks. This is important as messages are the cornerstone of data narrartives [15] and since the quality of notebooks is known to be very diverse [21].

## 3 FORMAL BACKGROUND

This section introduces the representation of EDA notebooks and messages and presents the set of properties used for learning interestingness.

## 3.1 Preliminary definitions

EDA notebooks are essentially sequences of programmatic operations and their commented results. They are linearly structured as a sequence of cells, of two types: text and code. Text cells contain explanatory text, typically including titles, definitions, explanations and comments. Code cells contain a sequence of commands and their output, typically including numeric results and graphics.

We consider that a code cell together with a text cell delivers a commented result on a logical part of the notebook. We will call it *message* in what follows. We represent a message as a pair of code and text cells, together with a set of numerical properties describing their contents (e.g. the number of words in the text cell, or the complexity of the code). The whole set of properties is described in next subsection.

*Definition 3.1 (Message).* Let $\mathcal{T}$ be an infinite set of text cells and $C$ an infinite set of code cells. A message is a tuple $m = \langle c, t, p_1^m, \ldots, p_o^m \rangle$ where $c \in C$ is a code cell, $t \in \mathcal{T}$ is a text cell and $p_i^m$, $1 \leq i \leq o$, are properties of $c$ or $t$.

| Dimension | Name | # |
|---|---|---|
| Notebook popularity | Number of likes | 0 |
| | Number of views | 1 |
| | Number of forks | 2 |
| | Author's expertise | 3 |
| Notebook structure | Number of cells | 4 |
| | Number of lines of code | 5 |
| | Number of lines text | 6 |
| Code cell | Number of characters | 7 |
| | Halstead score | 8 |
| | Cyclomatic complexity | 9 |
| | Generates a visualization | 10 |
| Text cell | Number of characters | 11 |
| | Number of words | 12 |
| | Flesch reading ease index | 13 |
| | Gunning-Fog index | 14 |
| | Automated Readability Index | 15 |
| | Coleman-Liau index | 16 |
| Message in notebook | Position in notebook | 17 |

**Table 1: Features considered**

Finally, we represent a notebook as a sequence of messages and a set of notebook properties (e.g. number of user's likes).

*Definition 3.2 (Notebook).* Let $\mathcal{M}$ be an infinite set of messages. A notebook is a tuple $n = \langle m_1, \dots, m_v, p_1^n, \dots, p_w^n \rangle$ where $m_i \in \mathcal{M}$, $1 \leq i \leq v$, are messages, and $p_j^n$, $1 \leq j \leq w$, are properties of $n$.

## 3.2 Properties

The properties of cells, messages and notebooks correspond to features extracted from notebooks, detailed in Table 1.

We consider the following feature dimensions:

- notebook popularity: these features indicate the global popularity of the notebooks among Kaggle users. They are the main drivers to compute messages' interestingness as they express the opinion of the community of users.
- notebook structure: these feature describe the size of the notebook in terms of cells and lines of code and comments.
- code cell: these features characterize code cells in terms of their complexity and the presence of a visualization. In addition to the number of lines of code, two classical software engineering metrics are used: cyclomatic complexity [13], Halstead metric [9].
- text cell: these features characterize the content of text cells especially in terms of readability, i.e., indexes related to the level of studies a person needs to understand the text at the first reading computed considering the number of words, number of sentences, number of syllables or number of characters as components.
- message characteristics: this feature indicates where the message is located in the notebook. Often the first messages of a notebook are simple data profiling while messages at the end tend to be more elaborated.

In the following we restrict to notebooks and messages over a given dataset. Implementation details about message and properties extraction are given in Section 5.

## 4 EXTRACTING NARRATIONS FROM NOTEBOOKS

In this section, we describe how we process notebook messages to extract narrations.

## 4.1 Problem definition

Let $M^D$ be the set of messages over a dataset $D$. We are interested in producing a sequence of $\epsilon_t$ messages from $M^D$ such that their total interestingness is maximal, and the overall cognitive distance between them is minimal.

This problem is defined formally in [2] as follows:

*Definition 4.1 (Traveling Analyst Problem (TAP)).* Let $Q$ be a set of $N$ queries, each associated with a positive time cost $cost(q_i)$ and a positive interestingness score $interest(q_i)$. Each pair of queries is associated with a metric $dist(q_i, q_j)$ representing the cognitive distance of browsing from one query result to the next. Given a time budget $\epsilon_t$, the optimization problem consists in finding a sequence $\langle q_1, \dots, q_M \rangle$ of queries, $q_i \in Q$, without repetition, with $M \leq N$, such that:

(1) $\max \sum_{i=1}^{M} interest(q_i)$
(2) $\sum_{i=1}^{M} cost(q_i) \leq \epsilon_t$
(3) $\min \sum_{i=1}^{M-1} dist(q_i, q_{i+1})$.

Lemma 4.2 (Complexity of TAP [2]). *TAP is strongly NP-hard.*

It can easily be seen that our problem is an instance of TAP, where queries are notebook messages and all their costs are the same. We next define the interestingness and distance functions.

## 4.2 Characterizing interestingness

To characterize the interestingness of messages, instead of proposing our own definition, we choose to learn a model of it, using the features of in Table 1. Our strategy is to compute a score for messages based on dimensions: Notebook popularity, Notebook structure and Message in notebook. And then, to learn this score using the features specific to messages, i.e., those in Dimensions Code cell and Text Cell.

We choose to focus on regression models as they give good results on similar problems [14]. We use auto-machine learning [7] to learn the model, since we aim to achieve good accuracy performances by testing a large spectrum of models and hyperparameters.

## 4.3 Ensuring relevance

Note that interestingness of messages is learned independently of any user requirement. In order to build a coherent data narrative in accordance with user interests, we introduce the notion of user profile and we propose to pre-filter the set of messages that are relevant to such a profile.

We model a user profile as a set of keywords representing user's interests. The relevance of a message for a user profile is computed based on the similarity of the text contained in the profile and in the text cell of the message. We use an off-the-shelf cosine similarity between the TF-IDF vectors of the user profile and the message. We use as document corpus the overall set of users' profiles $\mathcal{U}$ and text cells of all messages in $\mathcal{M}$.

Formally, let $m \in \mathcal{M}$ be a message, and $u \in \mathcal{U}$ be a user profile, with $t$ being the text cell of $m$. Let $V_1$ and $V_2$ be respectively the TF-IDF vectors of $u$ and $t$. The similarity between $u$ and $m$ is computed as:

$$sim(u, m) = cosine(V_1, V_2) \quad (1)$$

## 4.4 Characterizing distance

The distance between two messages is also computed based on the similarity of the text contained in the text cells. We use the same TF-IDF vectors for messages, computed for characterizing relevance.

Formally, let $m_1, m_2 \in \mathcal{M}$ be two messages, with $t_1$ and $t_2$ being respectively their text cells. Let $V_1$ and $V_2$ be respectively the TF-IDF vectors of $t_1$ and $t_2$. The distance between the two messages is computed as:

$$dist(m_1, m_2) = 1 - cosine(V_1, V_2) \tag{2}$$

## 4.5 Main algorithms

This subsection presents two algorithms that implement the approach. Algorithm 1 describes the extraction of messages, and the computation of their interestingness and distance. This algorithm can be executed offline. Algorithm 2 describes the generation of a data narrative for a specific user profile. It pre-selects relevant messages, calls the TAP for selecting and structuring messages and finally writes the narrative.

**Algorithm 1** Message extraction and computations

```
Require: a set of notebooks $N^D$, a set of user profiles $U$
Ensure: a set of messages $M^D$, an interestingness vector
    $interest$, a distance matrix $distance$
1: Let $M^D$ = extractMessages ($N^D$)
2: Let $interest()$ = learnInterestingness ($M^D, N^D$)
3: index ($M^D \cup U$)
4: Let $distance()$ = computeDistance ($M^D \cup U$)
5: return $M^D$, $interest()$, $distance()$
```

**Algorithm 2** User tailored narrative from notebook messages

```
Require: a set of messages $M^D$, a set of notebooks $N^D$, a user
    profile $u$, a similarity threshold $\epsilon_s$, a number of expected
    messages $\epsilon_t$
Ensure: a data narrative for the user
1: Let $M = \emptyset$
2: for $m \in M^D$ do
3:     if $sim(m, u) > \epsilon_s$ then
4:         $M = M \cup \{m\}$
5:     end if
6: end for
7: Let $T$ = TAP ($M, interest, distance, \epsilon_t$)
8: return narrate ($T, N^D$)
```

The *extractMessages* function extracts messages from a set of notebooks. Its implementation is described in Section 5. The *learnInterestingness* function computes message interestingness as described in Subsection 4.2. The *index* function indexes the corpus of messages and profiles, computing TF-IDF vectors, as described in Subsection 4.3. Such vectors are used for computing the distance among messages (*computeDistance* function) and similarity between a message and a profile (the *sim* function, which is $1 - distance()$). The TAP function implements the optimization problem described in Subsection 4.1. Finally, the *narrate* function generates the narration by writing messages in the order indicated by the TAP, reusing the original visualizations of messages.

## 5 IMPLEMENTATION AND TESTS

Our prototype is implemented in Python, using libraries Radon for code metrics and py-readability-metrics for readability metrics. We used Kaggle API to access the datasets and notebooks. To match a code cell with the visualization it produces, we used the HTML page of the notebook because the Kaggle API does not provide the visualization. We used Beautiful Soup to parse the HTML and mapped the visualization with the code cell using a join on the code text. We used sklearn for the TFIDF vectorization. Solving the TAP problem (see Section 4.1) exactly is done with a mathematical model on CPLEX 20.10 and is implemented in C++[2]. For large sets of messages (more than 500 messages) finding exact solutions is intractable. We use a fast and memory-efficient heuristics inspired by the classic "sort by item efficiency" heuristics for solving the Knapsack problem [4]. The code of the approach is available on Github[3].

We tested our code on 377 Kaggle notebooks from the first 18 datasets of Kaggle.com having more than 20 notebooks, sorted by votes. We extracted messages from these notebooks by considering only the code cells immediately followed by a text cell (a markdown cell in Kaggle terminology). This resulted in 10166 messages. The correlation matrix of the features of Table 1 is displayed in Figure 2, computed with Pearson's correlation coefficient. The order of features in the figure is the same as the order in the table. Globally it can be seen that the features are correlated when they are in the same feature dimension. In more details:

- in dimension notebook popularity, it can be seen that, unsurprisingly, likes, views and forks are quite correlated, while expertise is correlated to none of the others ;
- number of lines of codes and number of lines of text are only weakly correlated ;
- while code metrics are heavily correlated, they are not correlated to the length of the code neither and the generation of a visualization is correlated to none of the other features in this dimension ;
- interestingly, the position of messages is quite correlated to the total number of cells in the notebook and to the total number of lines of text lines, while it is less correlated to the number of lines of code. This reflects the correlations found in the notebook structure dimension and the fact that the more messages, the more cells in the notebook. On the other hand, the position of the message is not correlated to its own cells' code length or text length.

To learn interestingness, we use Auto-sklearn[4] with the principle presented in the previous section. Auto-sklearn produces an ensemble model that is not a single model but several models collaborating to achieve the best possible regression. The best ensemble model we obtained, in terms of $R^2$ is indicated in Table 2. Its $R^2$ score is 0.85 in the training phase and 0.59 in the testing phase. The target score we use was constructed by multiplying all the features in dimensions notebook popularity, notebook structure and message in notebook.

We created 20 user profiles by retrieving the owners of the datasets of Kaggle.com with the most votes and then retrieving all the datasets owned by these users. The words in the description of those datasets are used to form the profiles. For the 20 users, profiles ranged between 5 and 20 words, with an average of 14.5

[2]https://github.com/AlexChanson/Cplex-TAP
[3]https://github.com/Blobfish-LIFAT/NotebookCrowdsourcing
[4]https://automl.github.io/auto-sklearn/master/index.html

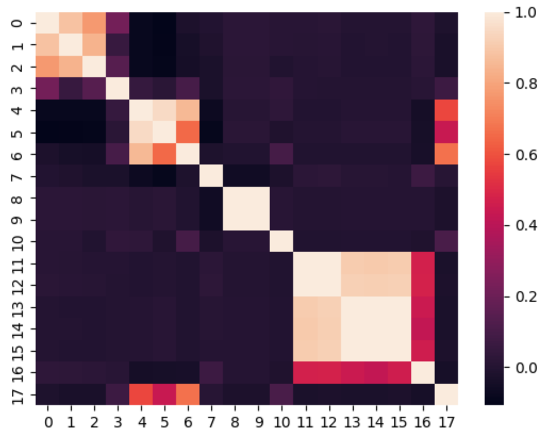

**Figure 2: Correlation of all the features in Table 1**

| rank | ensemble weight | type |
|---|---|---|
| 1 | 0.76 | gaussian process |
| 2 | 0.02 | gradient boosting |
| 3 | 0.04 | gradient boosting |
| 4 | 0.08 | k nearest neighbors |
| 5 | 0.10 | gradient boosting |

**Table 2: Model of interestingness**

(stdev is 4.97). The description of those datasets, together with the text of all text cells identified when extracting messages, formed the vocabulary from which TF-IDF vectors for users and messages were computed. For each user, we filtered the set of messages using their profile, using the cosine similarity between both TF-IDF vectors, using a threshold of 0. The number of messages relevant for each profile ranges between 191 (minimum) and 2551 (maximum), with an average of 798.

We generated one narration for each profile, asking for $\epsilon_t$=10 messages in it. On average, the generated narrations have 8.3 messages (minimum 2, maximum 10, stdev 1.75). To measure the degree of personalization of the narration, we use the Szymkiewicz–Simpson overlap coefficient[5] between the profile and the text of the messages. On average it is 0.15 (minimum 0.07, maximum 0.44, stdev 0.12). These low scores are expected since a threshold of 0 was used to select messages for each profile. To measure the coherence and diversity of the messages in the generated narrations, we measured (i) the number of different notebooks where the messages come from and (ii) the Szymkiewicz–Simpson overlap coefficient between the different message texts in the narration. Regarding (i), on average the messages come from 4.2 notebooks (minimum 1, maximum 9, stdev 0.3). As to (ii), the overlap is 0.68 on average (minimum 0.59, maximum 0.77, stdev 0.04). The generated narrations, under the form of Jupyter notebooks, are available on Github[6].

[5]The overlap coefficient is defined as the size of the intersection divided by the smaller of the size of the two sets. It is a form of Jaccard coefficient adapted to sets with different cardinality.
[6]https://github.com/Blobfish-LIFAT/NotebookCrowdsourcing/tree/master/output/notebooks

## 6 CONCLUSION

This short paper introduces a novel approach for generating personalized data narratives from EDA notebooks. The approach consists of extracting messages from existing notebooks, learning their interestingness, filtering this set of messages for some user profile and generating a coherent sequence of messages adapted to this profile. We detailed the implementation of our proof of concept, and presented a preliminary experiment with Kaggle.com notebooks.

We are currently working at improving our approach by providing more robust message detection, better accounting for the visualizations related to the message, generating narratives that are more coherent, less redundant and more personalized. We will evaluate the approach with user tests, comparing it with competitor approaches to generate notebooks [6, 16] and assessing its scalability.

## REFERENCES

[1] Sheelagh Carpendale, Nicholas Diakopoulos, Nathalie Henry Riche, and Christophe Hurter. Data-driven storytelling (dagstuhl seminar 16061). *Dagstuhl Reports*, 2016.
[2] Alexandre Chanson, Ben Crulis, Nicolas Labroche, Patrick Marcel, Verónika Peralta, Stefano Rizzi, and Panos Vassiliadis. The traveling analyst problem: Definition and preliminary study. In *DOLAP@EDBT/ICDT*, 2020.
[3] S. Chen, J. Li, G. Andrienko, N. Andrienko, Y. Wang, P. H. Nguyen, and C. Turkay. Supporting story synthesis: Bridging the gap between visual analytics and storytelling. *TVCG*, 2018.
[4] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–288, 1957.
[5] Daniel Deutch, Amir Gilad, Tova Milo, and Amit Somech. Explained: Explanations for EDA notebooks. *Proc. VLDB Endow.*, 13(12):2917–2920, 2020.
[6] Ori Bar El, Tova Milo, and Amit Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *SIGMOD*, 2020.
[7] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems, Canada*, 2015.
[8] Dimitrios Gkesoulis, Panos Vassiliadis, and Petros Manousis. Cinecubes: Aiding data workers gain insights from OLAP queries. *Inf. Syst.*, 53:60–86, 2015.
[9] Maurice H. Halstead. *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc., USA, 1977.
[10] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, 2015.
[11] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *IEEE Computer*, 46, 2013.
[12] John McAuley, Rohan Goel, and Tamara Matthews. Explorobot: Rapid exploration with chart automation. In *VISIGRAPP*, 2019.
[13] Thomas J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.
[14] Martina Megasari, Pandu Wicaksono, Chiao Yun Li, Clément Chaussade, Shibo Cheng, Nicolas Labroche, Patrick Marcel, and Verónika Peralta. Can models learned from a dataset reflect acquisition of procedural knowledge? an experiment with automatic measurement of online review quality. In Il-Yeol Song, Alberto Abelló, and Robert Wrembel, editors, *Proceedings of DOLAP*, volume 2062 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
[15] Faten El Outa, Matteo Francia, Patrick Marcel, Verónika Peralta, and Panos Vassiliadis. Towards a conceptual model for data narratives. In *ER*, 2020.
[16] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Équille, Maximilian Fabricius, and Srividya Subramanian. DORA THE EXPLORER: exploring very large data with interactive deep reinforcement learning. In *CIKM*, 2021.
[17] Adam Rule, Aurélien Tabard, and James D. Hollan. Exploration and explanation in computational notebooks. In *CHI*, 2018.
[18] D. Shi, F. Sun, X. Xu, Xingyu Lan, David Gotz, and Nan Cao. Autoclips: An automatic approach to video generation from data facts. *Comput. Graph. Forum*, 40(3):495–505, 2021.
[19] Danqing Shi, Xinyue Xu, Fuling Sun, Yang Shi, and Nan Cao. Calliope: Automatic visual data story generation from a spreadsheet. *IEEE Trans. Vis. Comput. Graph.*, 27(2):453–463, 2021.
[20] Bo Tang, Shi Han, Man Lung Yiu, Rui Ding, and Dongmei Zhang. Extracting top-k insights from multi-dimensional data. In *SIGMOD*, 2017.
[21] Jiawei Wang, Li Li, and Andreas Zeller. Better code, better sharing: on the need of analyzing jupyter notebooks. In *ICSE-NIER*, 2020.
[22] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. Datashot: Automatic generation of fact sheets from tabular data. *IEEE Trans. Vis. Comput. Graph.*, 2020.