DescribeML: A Tool for Describing Machine Learning Datasets

Joan Giner-Miguelez
Internet Interdisciplinary Institute, IN3
Universitat Oberta de Catalunya, UOC
Barcelona, Spain
jginermi@uoc.edu

Abel Gómez
Internet Interdisciplinary Institute, IN3
Universitat Oberta de Cataluny, UOC
Barcelona, Spain
agomezlla@uoc.edu

Jordi Cabot
ICREA &
Internet Interdisciplinary Institute, IN3
Universitat Oberta de Cataluny, UOC
Barcelona, Spain
jordi.cabot@icrea.cat

ABSTRACT

Datasets play a central role in the training and evaluation of machine learning (ML) models. But they are also the root cause of many undesired model behaviors, such as biased predictions. To overcome this situation, the ML community is proposing a *data-centric cultural shift*, where data issues are given the attention they deserve, for instance, proposing standard descriptions for datasets.

In this sense, and inspired by these proposals, we present a model-driven tool to precisely describe machine learning datasets in terms of their structure, data provenance, and social concerns. Our tool aims to facilitate any ML initiative to leverage and benefit from this data-centric shift in ML (e.g., selecting the most appropriate dataset for a new project or better replicating other ML results). The tool is implemented with the Langium workbench as a Visual Studio Code plugin and published as an open-source.

1 INTRODUCTION

Motivation: As data is gaining centrality, especially in machine learning (ML) applications, the processes involved in building by datasets are becoming more complex [8]. Dataset creation involves different teams and stages such as gathering, labeling, and design. Recent studies have reported undesired consequences, and negative downstream effects in the whole machine learning pipeline due to data issues [13, 14]. For example, facial analysis datasets with a low number of darker-skinned faces could drop the accuracy of face analysis models in that particular group, representing social harm to them [9]. As another example, a natural language dataset gathered from Australian speakers could drop the accuracy of models trained to support users of the United States due to the different language styles [1]. In both examples, we see the need to store information about the gathering and labeling stages, or high-level analysis, such as the social impact on specific groups.

State-of-the-art: This situation has brought recent interest inside the research community about a *data-centric cultural shift* in the machine learning field[1]. The standardization of data creation processes, the need for formal documentation, and the need for mature tools to adopt best practices are common demands inside the research community. Therefore, recent works such as *Datasheets for datasets*, among others [1, 3, 4, 7, 11], have proposed the main guidelines for the creation of standard documentation for datasets. In these proposals, the authors pinpoint the data aspects that could affect how the dataset should be used, or the quality of the ML models trained with it.

[1]https://spectrum.ieee.org/andrew-ng-data-centric-ai

Our proposal: In this sense, we present DescribeML, a model-driven tool to precisely describe datasets according to the dimensions demanded by the aforementioned proposals. This tool provides a specific notation based on a domain-specific language (DSL) described here [5], and guides creators through the dataset description process with common modern language features such as auto-completion, syntactic and semantic highlight, validation, cross-references etc.

Moreover, once the dataset is modeled using our tool, it can then be easily manipulated with model-driven engineering tools and techniques, opening the door to a number of (semi)automated application scenarios. For instance, searching the most suitable dataset based on the requirements of the ML projects (e.g., searching for a dataset compliant with specific social concerns, such as specific demographic), starting what, in a future, could become a dataset marketplace. We developed *DescribeML* using the Langium language workbench as a plugin for Visual Studio Code. The plugin is open-sourced in a public repository[2] and released in the Visual Studio Code Marketplace

The paper is structured as follows: Section 2 presents the DSL behind our proposal notation, Section 3 presents the architecture and the development process of the tool, Section 4 presents the usage of the tool and, finally, Section 5 wraps up the conclusions.

2 MODELING MACHINE LEARNING DATASETS

To implement our tool, we used a domain-specific language (DSL) proposal intended to describe dataset for machine-learning [5]. This language, inspired by the recent documentation proposals in the ML field, offers a set of modeling primitives to describe the different stages (gathering, labeling, design, etc.) of the dataset creation, and relevant aspects that may affect the uses of the data and the quality of the models using these data. The DSL is structured in four main parts (*Metadata*, *Composition*, *Provenance*, *SocialConcerns*) that can be seen in Figure 1.

In the *Metadata* part, we can express the general information about the dataset such as the *uniqueId*, the *description*, the *dates* (update, release, published) and the associated *tags*. In parallel, we can express information regarding the uses of the data, such as *licenses* and recommended and non-recommended *applications*. Finally, we can set the *authoring* information, such as the authors, funders, or the maintenance policies of the dataset.

In the *Composition* part we can express aspects concerning the data structure, statistical description values, quality metrics, and the consistency rules that the dataset satisfies. As we can see in

[2]https://github.com/SOM-Research/DescribeML

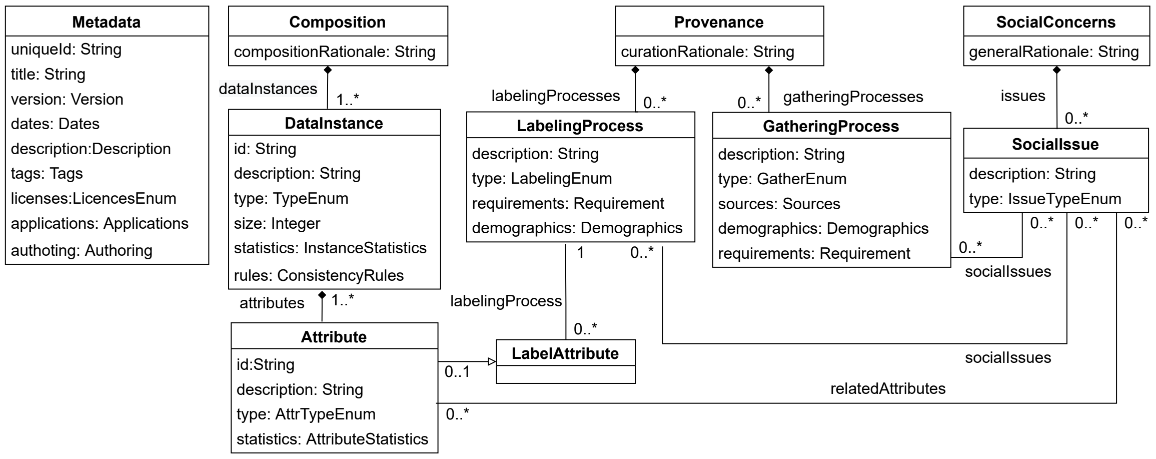


Figure 1: DSL concepts overview

Figure 1, the DSL allows defining a set of data instances [3] and the attributes composing these instances. At instance level, we can express the *description*, the *type* (e.g. tabular data), and the *size* together with the instance *statistics* such as the number of missing values or specific correlation between attributes. Finally, a set of consistency *rules*, using the Object Constraint Language (OCL) [2], can be expressed. For each *Attribute*, we can express the *type* of the attribute (such as categorical or numerical), relate the attribute with specific labeling process if it is a label, and express relevant *statistics* of the attribute (e.g. the length, the mean, or the categorical distribution).

In the *Provenance* part, we can express aspects about the gathering and labeling process such as the *description* of the process, the process *type* (e.g. image annotation, or manual human curators), the *requirements* of each process and information regarding the data sources. In addition, we can express the *demographics* of the processes (information about who labeled the data, and who gathered the data). Finally, the *Social Concerns* part, allows expressing specific social issues of the data (such as bias, data potential harms, or privacy) and relate these issues with specific labeling and gathering processes (e.g. the labeling team may be biased), or specific attributes (e.g. "patients_ID" has a privacy issue).

3 ARCHITECTURE AND DEVELOPMENT

We have implemented the DSL presented in the last section as a plugin for Visual Studio Code. We have chosen the Visual Studio Code (VSCode) environment since it is one of the most popular development environments in the machine learning field. To implement the tool, we have used Langium[4] and the VSCode extension API[5]. Langium is a low-code language engineering toolkit based on Typescript for Visual Studio Code to create textual DSLs. From a particular grammar, Languim generates a language plugin for VSCode with modern language features such as autocomplete, syntactical highlight and cross-references.

In parallel, the VSCode extension API, acts as a wrapper of the Language Server Protocol allowing us to develop advanced language services (beyond the provided by Langium out-of-the-box), and also, allowing us to develop custom IDE integration services such as the developed data preloader or the documentation generator of our tool explained in the sections below.

In Figure 2, we can see an overview of the development process of the tool. At first, we have defined a *Grammar* as a specific notation of the DSL using the Langium Grammar Language[6]. Then, Langium takes this declarative grammar and generates a *language plugin* for VSCode in Typescript. Then we have developed a set of custom services in Typescript using the Visual Code extension API, and we have integrated them with the generated language plugin. Finally, we compile the code with a Typescript compiler, and we get a *VSCode plugin* ready to be used in the IDE.

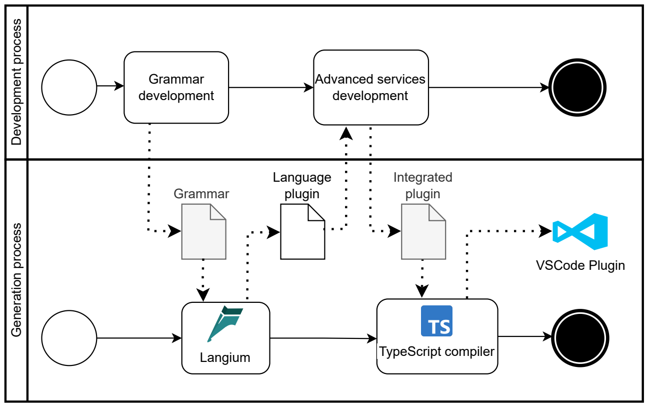


Figure 2: Tool development process overview

[3]Notice that, in the data science field, an *instance* is understood as the group of attributes of an entity in the real world, similarly to the concept of *class* in the modeling community and therefore radically different from our typical understanding of the word *instance* in object-oriented programming.
[4]https://langium.org/
[5]https://code.visualstudio.com/api

3.1 The Grammar

The first step of the process has been the *Grammar Development*. We have implemented the DSL metamodel to a specific grammar notation, combining the Langium grammar language and the Extended Backus–Naur Form (EBNF) syntax [10]. The full grammar of the tool can be seen at the tool open-source repository [7].

In Listing 1, we can see an excerpt of the defined grammar. Using the notation supported by Langium, we have expressed the optional attributes using the "?" symbols, such as in line 9, where *citation* is an optional attribute. Besides, the "|" symbol acts as an OR operator, so in line 10, after the *Description* keyword, we give the option to provide a single string as the description or a three-folded structure with the purposes, the tasks, and the gaps of the dataset. Also, to express zero to any multiplicity relation, we use the "*" symbol. For instance, in line 13, you can provide a set of machine-learning tasks.

In addition, some keywords are referencing enumerates such as *CommonLicenses* and *MLTasks*, in lines 33 and 36, that compiles the common licenses and tasks in the machine learning field. Finally, we have expressed the cross-references using brackets such as in line 29, where we can assign a set of related *Attributes* to a specific *Social Issue*. As an example, an attribute "gender" may be related with a gender parity social issue.

[6]https://langium.org/docs/grammar-language/
[7]https://github.com/SOM-Research/DescribeML/blob/main/src/language-server/dataset-descriptor.langium

Listing 1: Grammar excerpt

```
1  Metadata:
2       name= 'Metadata:'
3       'Title:' title=STRING
4       'Unique-identifier:' ident=ID
5       'Version:' version=ID
6       ('Release Date:' dateRe=DateYear )?
7       ('Updated Date:' dateUp=DateYear )?
8       ('Published Date:' datePu=DateYear )?
9       (citation=Citation)?
10      'Description:'
11         (description=STRING |
12         (('Purposes:' descriptionpurpose=STRING)?
13          ('Tasks:' '[' descriptionTasks+=MLTasks
14                     ((','descriptionTasks+=MLTasks)*']')?)?
15          ('Gaps:' descriptionGaps=STRING)?))
16      ('Licences:' licence=CommonLicences)?
17      (uses=Applications)?
18      (distribution=Distribution)?
19      (area=Area)?
20      (tags=Tags)?
21       authoring=Authoring;
22
23 Attribute:
24     //...
25
26 SocialIssue:
27     'Social Issue:' name=ID
28     'IssueType:' IssueType=SocialIssueType
29     ('Related Attributes:'('attribute:'rAtt=[Attribute])*)?
30     'Description:' desc=STRING
31     //...
32
33 CommonLicences returns string:
34     'Creative Commons' | 'CC0: Public Domain' | //...
35
36 MLTasks returns string:
37     'text-classification'|'question-answering'| //...
```

3.2 Advanced Services

Once we developed the grammar, and we had the first version of the language VSCode plugin generated by Languim, we developed a set of custom services using the VSCode extension API. Languim instantiates the grammar as an Abstract Syntax Tree (AST), which is represented as a collection of TypeScript types describing the content of a parsed document hierarchically, where the nodes are represented as a JavaScript objects. Using these objects, and the wrapper services provided by the Visual Studio Extension API, we developed a set of custom services to complement our plugin. We develop two types of custom services, the services providing custom language features, and the services enhancing the IDE capabilities.

Advanced language features: We override the custom validation service to add custom validation and hints during the documentation process. The hint of the documentation process are based on the documentation guidelines provided by recent works [1, 3, 4, 7, 11] in the ML field. These hints provide context to dataset creators, facilitating the consistency and the correct usage of the description features.

To implement semantic highlight, VSCode uses the TextMate grammar as the tokenization engine. This allows us to provide extra information during the highlighting process based on the language knowledge. In our case, we mapped the default tokens provided by the VSCode extension API, such as class, keywords, and variables with the classes, attributes, and attribute's values of our grammar.

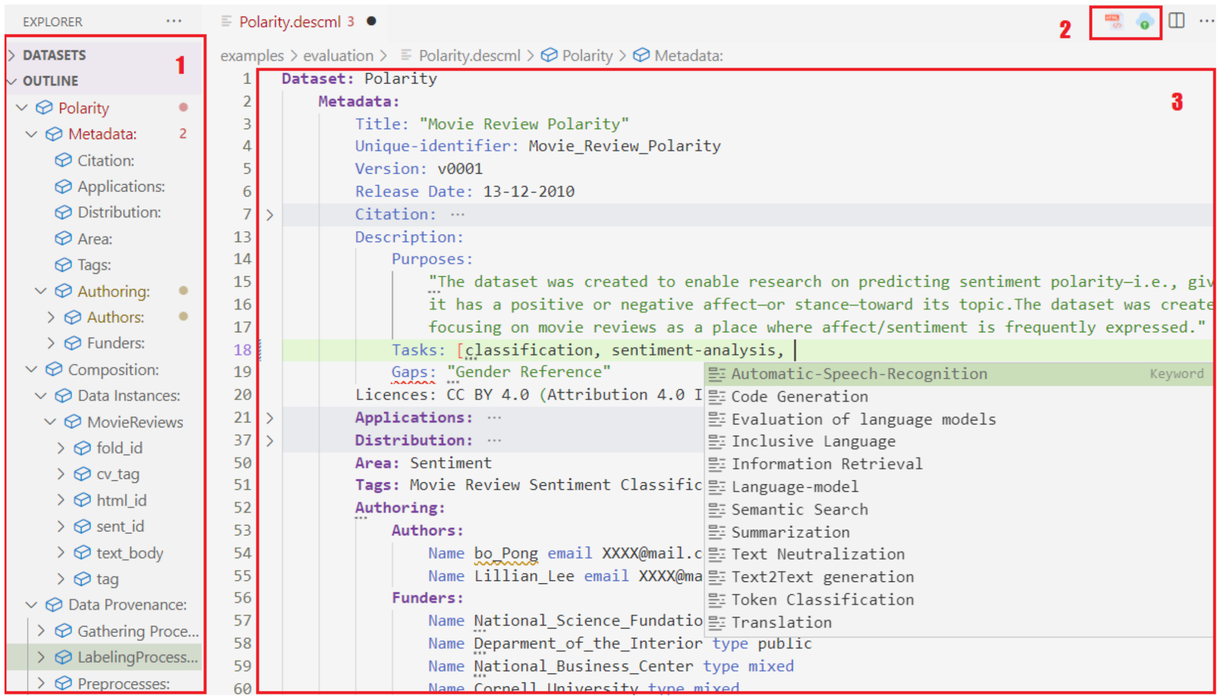
Enhancing IDE capabilities: We developed a data preloader service to upload data files and create, automatically, the first draft of the description document. This service gets the .csv files containing the data and apply a set of heuristics to extract basic data information such as title, date, instances and attributes structure, and basic statistics.

Finally, we create a document generator that takes a valid document description and generates HTML documentation. This HTML documentation is also populated with the Schema.org vocabulary [6], a vocabulary to facilitate the discoverability of the content by the search engines. In particular, we implement the extension *"@dataset"* of the vocabulary, as this is used by the Google Dataset Search engine[8] to discover datasets across the web.

4 TOOL USAGE

One of the main goals when designing our tool has been to keep it simple and easy to use. Therefore, the installation process is as

[8]https://datasetsearch.research.google.com/



```
Dataset: Polarity
    Metadata:
        Title: "Movie Review Polarity"
        Unique-identifier: Movie_Review_Polarity
        Version: v0001
        Release Date: 13-12-2010
        Citation: ···
        Description:
            Purposes:
                "The dataset was created to enable research on predicting sentiment polarity—i.e., giv
                it has a positive or negative affect—or stance—toward its topic.The dataset was create
                focusing on movie reviews as a place where affect/sentiment is frequently expressed."
            Tasks: [classification, sentiment-analysis, |
            Gaps: "Gender Reference"
        Licences: CC BY 4.0 (Attribution 4.0 I
        Applications: ···
        Distribution: ···
        Area: Sentiment
        Tags: Movie Review Sentiment Classific
        Authoring:
            Authors:
                Name bo_Pong email XXXX@mail.c
                Name Lillian_Lee email XXXX@ma
            Funders:
                Name National_Science_Fundatio
                Name Deparment_of_the_Interior type public
                Name National_Business_Center type mixed
                Name Cornell_University type mixed
```

```
Automatic-Speech-Recognition          Keyword
Code Generation
Evaluation of language models
Inclusive Language
Information Retrieval
Language-model
Semantic Search
Summarization
Text Neutralization
Text2Text generation
Token Classification
Translation
```

Figure 4: Tool editor overview

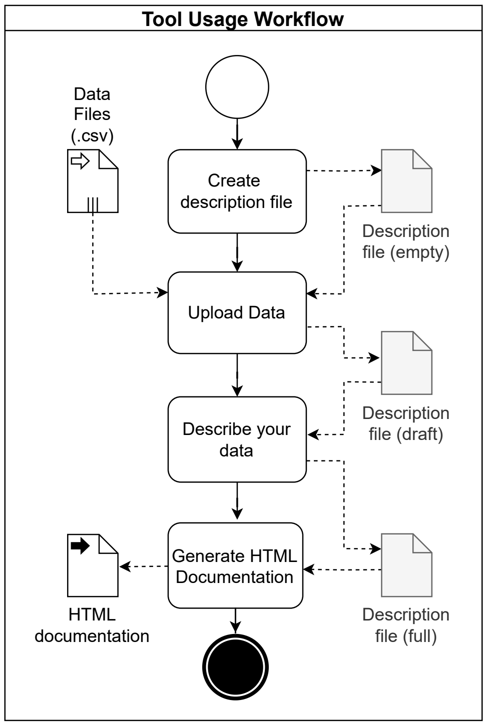


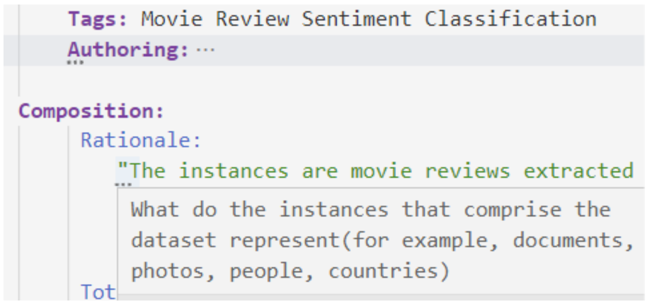
Figure 3: Tool usage workflow

simple as searching the plugin in the VSCode extension tab, download and enable it. Alternatively, the plugin can also be installed by getting the extension file (.vsix) published in the root folder of the public repository. Then, the main usage workflow of the tool is composed of four stages, shown in Figure 3. At first, we need to create a description file (.descml) to allow the IDE to recognize it. Then we have to preload our data files to fill, automatically, the created document with a first draft of the description containing the extracted information from the data (currently, only .csv is supported). Then, helped by the present language features, we need to fill the description document, and finally, once we have a full valid document, we can generate the HTML documentation.

In terms of UI, we have chosen to keep the developer's experience similar to developing code with VSCode. We chose this approach as we believe the IDE developer experience is already familiar for developers of the ML community and can help to flatten the learning curve of our tool. In Figure 4, we can see an overview of the editor's tool. In the square marked with the number "1", we see the representation of the instantiated Abstract Syntax Tree as an outline of the description. In the square marked with the number "2" we have the two action buttons implementing the custom data preloader and document generation services. Finally, in the square market with the number "3", we have the editor. As we see, the editor provides features such as syntactic and semantic highlight, auto-completion, and validation during the dataset's description process.

As an example of usage of the tool, we have released a set of descriptions of popular datasets in a public repository[9]. We chose

[9]https://github.com/SOM-Research/DescribeML/tree/main/examples/evaluation



```
    Tags: Movie Review Sentiment Classification
    Authoring: ···

Composition:
    Rationale:
        "The instances are movie reviews extracted
         What do the instances that comprise the
         dataset represent(for example, documents,
         photos, people, countries)
    Tot
```

Figure 5: Tool's hints feature example

these datasets based on the fact that they were already the target of some of the mentioned recent works in the ML community [3, 4, 11] about dataset documentation practices, and/or have a diverse provenance and composition. We believe that these descriptions can be useful as a way to demonstrate our tool.

In Figure 4 we are describing the *Movie Reviews Polarity* [12] dataset, a wide-use benchmark dataset for sentimental analysis tasks, composed of a set of movie reviews tagged with a sentimental flag (such as positive, negative) by a group of reviewers. In the figure, we can see the description of the metadata part. In line 17 of the text editor, we see the auto-completion feature giving suggestions to the user regarding the machine learning tasks this dataset is intended for. Moreover, we can see the *Area* and *Tags* for the dataset together with the authors and funders. In addition, every attribute marked with the horizontal points, such as line 14, 18, 51, 56, and 57 provides hints to assist creators during the description process. In Figure 5, we can see an example of one of these hints for the *Rationale* attribute of the *Composition part*.

5 CONCLUSIONS

In this paper, we have presented a tool for describing datasets for the Visual Studio Code environment. The tool assists practitioners during the dataset description using the presented DSL [5]. We believe this tool is a step forward towards the standardization of dataset descriptions and its future impact in achieving higher quality ML models, especially from a social perspective (fairness, diversity, absence of bias, etc.).

As future work, and following the Langium roadmap, we plan to adapt the tool to web browsers, to be able to integrate our tool in any web app without the need of having a full IDE instance running in the back end. On the other hand, we plan to validate the tool with end-users from the ML community in production environments.

ACKNOWLEDGEMENTS

The research described in this paper has been partially supported by the Spanish government (LOCOSS - PID2020-114615RB-I00), and the AIDOaRT Project, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

REFERENCES

[1] Emily M. Bender and Batya Friedman. 2018. Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics* 6 (2018), 587–604.
[2] Jordi Cabot and Martin Gogolla. 2012. Object constraint language (OCL): a definitive guide. In *International school on formal methods for the design of computer, communication and software systems*. Springer, 58–90.
[3] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.
[4] Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672* (2021).
[5] Joan Giner-Miguelez, Abel Gómez, and Jordi Cabot. 2022. A domain-specific language for describing machine learning datasets. https://doi.org/10.48550/ARXIV.2207.02848
[6] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: evolution of structured data on the web. *Commun. ACM* 59, 2 (2016), 44–51.
[7] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. 2020. The dataset nutrition label. *Data Protection and Privacy, Volume 12: Data Protection and Democracy* 12 (2020), 1.
[8] Ben Hutchinson, Andrew Smart, Alex Hanna, Emily Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. 2021. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 560–575.
[9] Ashraf Khalil, Soha Glal Ahmed, Asad Masood Khattak, and Nabeel Al-Qirim. 2020. Investigating bias in facial analysis systems: A systematic review. *IEEE Access* 8 (2020), 130751–130761.
[10] Daniel D McCracken and Edwin D Reilly. 2003. Backus-naur form (bnf). In *Encyclopedia of Computer Science*. 129–131.
[11] Angelina McMillan-Major, Salomey Osei, Juan Diego Rodriguez, Pawan Sasanka Ammanamanchi, Sebastian Gehrmann, and Yacine Jernite. 2021. Reusable Templates and Guides For Documenting Datasets and Models for Natural Language Processing and Generation: A Case Study of the HuggingFace and GEM Data and Model Cards. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics*. ACM, Online, 121–135.
[12] Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. 271–es.
[13] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M Bender, Emily Denton, and Alex Hanna. 2021. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns* 2, 11 (2021), 100336.
[14] Cedric Renggli, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlaš, Wentao Wu, and Ce Zhang. 2021. A Data Quality-Driven View of MLOps. *Data Engineering* (2021), 11.