How COVID-19 Impacted Data Science: a Topic Retrieval and Analysis from GitHub Projects' Descriptions

Amanda C. R. Tavares, Natércia A. Batista, Mirella M. Moro

[1] Universidade Federal de Minas Gerais – Belo Horizonte, Brazil

amandacrisrodriguest@gmail.com, {natercia,mirella}@dcc.ufmg.br

Abstract. *We present a data-driven research over code repositories that are data science oriented. The goal is to compare their topics of interest and evolution over the COVID-19 pandemic period by analyzing Jupyter Notebook and Python projects from a year before and during the pandemic. We employ a state-of-art algorithm to find topics based on the repositories descriptions, and compare the performance of tuning its hyperparameters for better accuracy.*

1. Introduction

Since its first outbreak in December 2019, COVID-19 pandemic has put the world through unprecedented situations. Specially in Computer Science, recent studies go over the effects on software developers and their projects [Ralph et al. 2020, Silveira et al. 2021], and characterize COVID-19 GitHub projects [de Oliveira et al. 2021, Wang et al. 2020]. Nevertheless, few studies focus on the impacts of the pandemic on the Data Science field.

Data Science and its two main components (Artificial Intelligence – AI, and Machine Learning – ML) have recently become popular for collaborative development and research. In GitHub,[1] the most used language for AI and ML is Python, followed by the interactive literate programming paradigm of Jupyter Notebook [Gonzalez et al. 2020]. The popularity of Jupyter Notebook among data scientists can be explained by its facilities to document, modify, and experiment with data [Perkel 2018, Pimentel et al. 2021].

Our work aims to fill such a gap by analyzing data science-oriented coding repositories and comparing their topics pre and during the pandemic. Our novel methodology includes: building a dataset of near 60 thousand GitHub repositories in Python or Jupyter Notebook; finding their topics by using Latent Dirichlet Allocation (LDA); tuning LDA after an extensive analysis of six metaheuristics to get the fittest models; and analyzing such fitness according to a coherence measure (instead of the usual Silhouette coefficient [Sharma et al. 2017]). Note that considering the repository own topics is not enough, as its users manually assign them, making it an error-prone task [Sipio et al. 2020].

2. Related Work

This section summarizes recent work on topic classification techniques and characterization over data science and COVID-19 related GitHub repositories.

Topic Classification. Saraiva and Medeiros (2018) introduce the CIMAL framework that finds relationships between topics in educational material from textual sources, slides,

[1] GitHub is the largest collaborative software development platform with over 65 million developers working on 200+ million repositories using different programming languages.

and videos. It uses Explicit Semantic Analysis (ESA) for topic classification by correlating relatedness between textual parts. Sharma et al. (2017) extract README files and descriptions from 10,000 randomly selected popular GitHub repositories to identify categories by using a combination of LDA and Genetic Algorithm (LDA-GA) modeling. They also performed manual analysis to generate categories based on the LDA topics.

GitHub and COVID-19. Wang et al. (2020) consider over 67,000 COVID-19 related projects, then manually analyze only the top 200 repositories to identify six categories. The classification uses the cosine similarity of the TF-IDF vectorization from all descriptions, compared to manually categorized training data consisting of 100 repositories for each category. Likewise, de Oliveira et al. (2021) consider 60,352 GitHub repositories and 1,190 questions from Stack Overflow and Data Science Q&A related to COVID-19. They also model the topics by using the LDA algorithm, then identifying eleven topics from the descriptions of the GitHub projects and seven from the Q&A forums.

Contributions. There is no publication on data-driven research that evaluates data science oriented GitHub repositories during the pandemics. In this initial work, we implement a topic modeling and analysis by applying LDA combined with unsupervised optimization for its hyperparameter tuning over the repositories' description (different from [Sharma et al. 2017] that uses README files). For such tuning, we compare six metaheuristics and select the best performing one (as inspired by [Panichella 2021]). For fitness function, we use the c_v coherence measure of the model, as it consumes less computer memory and processing time compared to the Silhouette coefficient [Sharma et al. 2017], based on our pre-analyses. The c_v coherence measure is one of the best performing coherence evaluation methods as it is strongly correlated to human ratings [Röder et al. 2015], meaning that a model with a higher c_v score has topics with superior quality.

3. Methodology

To analyze and compare data science projects using Jupyter Notebooks and Python, we introduce a methodology with: data extraction, data preparation, synthesis, and analysis.

Data Extraction. The first step uses a data crawler and the GitHub REST API[2] to get projects with Jupyter Notebook and Python as programming languages and at least six stars created in 2019 and 2020. The number of stars limits the search to repositories with minimum relevance to the community, while still containing a general representation for each language and not just the most popular repositories [Gonzalez et al. 2020]. As a secondary filter, we classify the descriptions idiom using the package pylang[3] and get only English-written projects without empty text; then removing up to 40% of the repositories.

Data Preparation. Preparing the data requires: removing URLs, emojis, non-ASCII characters, punctuation, and stop words from the texts; performing word lemmatization using the package spaCy[4]; filtering identified non-software projects based on specific keywords (e.g., "course", "tutorial"); detecting bigrams using gensim[5] Phrase (collocation) detection; and grouping bigram terms that appear at least five times through all documents for Jupyter Notebook and at least ten times for Python. Then, we calculate the TF-IDF

[2]GitHub REST API: `https://docs.github.com/en/rest`
[3]pylang: `https://pypi.org/project/pylang/`
[4]spaCy: `https://spacy.io/`
[5]gensim: `https://radimrehurek.com/gensim/index.html`

Table 1. Statistics of the final dataset

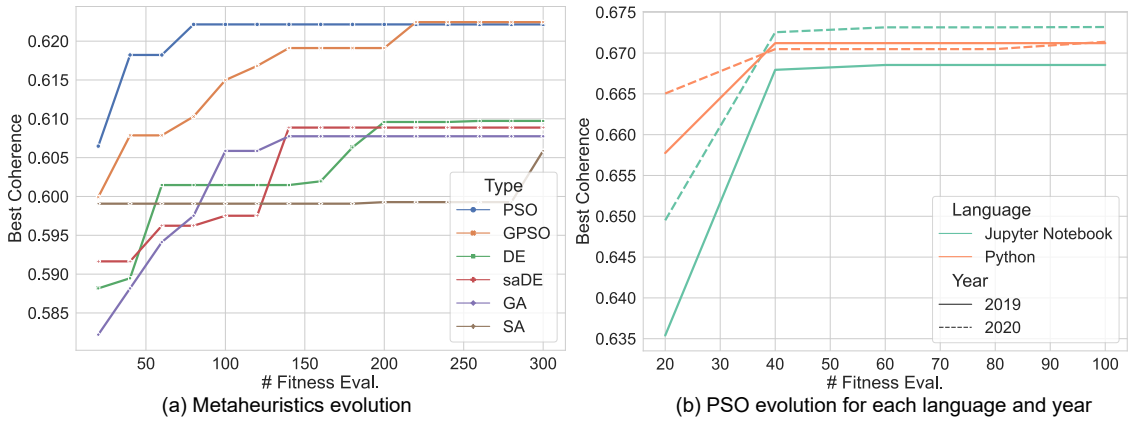
	2019		2020	
Language	**Original**	**Final**	**Original**	**Final**
Jupyter Notebook	10,278	4,891	9,120	4,088
Python	43,731	26,644	39,229	23,288

(Term Frequency and Inverse Document Frequency) score of the words with the gensim implementation to filter out very common words (low TF-IDF) that do not bring value (e.g., "data") and uncommon ones (high TF-IDF) that could be irrelevant or even typos (e.g., "alogrithm" – sic). Initial analyses revealed values smaller than 0.2 or larger than 0.8 for such words; hence, those words were filtered out. Lastly, we consider the number of words within a repository description. At this stage, an analysis informed most descriptions have between two and eight words. To normalize by size, descriptions out of such range were filtered out (normalizing by size seeks better results for topic classification). To identify topics, LDA requires as input: a dictionary of terms (i.e., term and its frequency over all documents) and the documents as bag of words, both of which are created from the descriptions at this point. To improve the results, we also clean the dictionary by removing terms that appear in less than five documents for Jupyter Notebook and less than ten for Python to exclude infrequent words, as Python has double the words of Jupyter in our final dataset. The dataset consists of 58,911 repository descriptions separated by language and creation year, according to Table 1 and available at [Tavares et al. 2021].

Synthesis. After preparing the data, we may identify the topics from the repositories descriptions. For topic modeling, we apply the LDA algorithm with the number of passes set to 10 and iterations to 100 for the model, as those are correlated with the number of documents to be analyzed based on our testings. LDA also requires setting up near optimal values for its parameters: k reflects the number of topics for the model, α sets the topic density for each document, and β regards the word density in the topic. Untuned LDA leads to underperforming classification results. Hence, using metaheuristics algorithms helps to optimize the tuning process by automatically finding near-ideal values for each hyperparameter. Inspired by [Panichella 2021], we test six of such algorithms over 10,000 descriptions of randomly selected Jupyter Notebook repositories: Differential Evolutionary (DE) and its Self Adaptive version (SADE), Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and its Generational version (GPSO) and Simulated Annealing (SA). SA is the only local search metaheuristic used, while the others (DE, SADE, GA, PSO, GPSO) are population based. As aforementioned, we evaluate them with the c_v score. For a fair comparison, we set their stopping criterion to the same number of fitness evaluations (FE) $= 300$ and use a fixed random seed value (0). The value range for the hyperparameters are $k = [10, 30]$, $\alpha = [0.001, 0.5]$ and $\beta = [0.001, 0.5]$. After comparing the results from the metaheuristic algorithms, we select the best according to c_v score and run it for each language and creation year with the value of FE for which the results from the previous test converged (as presented in Section 4). We use gensim implementation of the LDA and pygmo[6] framework to build our parameter tuning optimization for the LDA.

Analysis. After running LDA, we analyze its resulting topics to define their domain

[6]pygmo: `https://esa.github.io/pygmo2`



(a) Metaheuristics evolution (b) PSO evolution for each language and year

Figure 1. Coherence Values Evolution

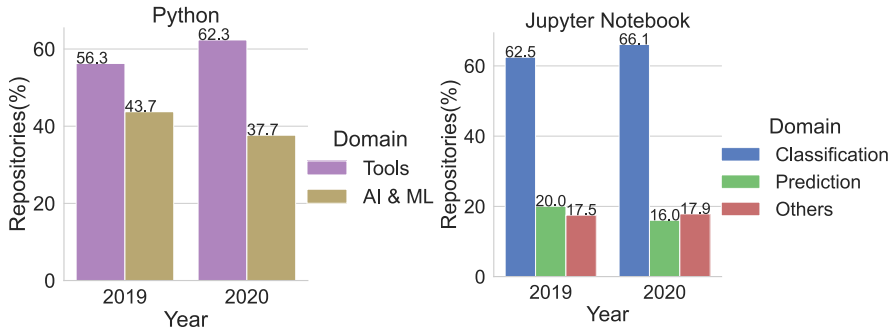


Figure 2. Percentage of repositories for each domain, language and year

by creating word clouds for each topic. Note that we create four LDA models: one for each language/year pair. We also select the top 50 descriptions for each topic to help our manual topic labeling and the domain classification. The next section sums up the results.

4. Results and Discussion

This section presents results in three parts: (*i*) algorithm performance for choosing the fittest LDA parameters according to the coherence measure, (*ii*) the domains and their distributions for each language/year, (*iii*) and topic analysis for each domain/year.

LDA tuning algorithms performance. Figure 1 (a) shows the coherence score for each algorithm: PSO and GPSO provide the best results; although PSO converged in fewer fitness evaluations than GPSO. Hence, we use PSO to tune the LDA parameters for the descriptions of each language and year. Figure 1 (b) presents the best coherence score values regarding FE specifically for PSO algorithm. As the best coherence was achieved with $FE = 100$ for most language/year pairs, such value is used as stop criterion.

Domain distribution. From the topics for each language, we may identify two domains for Python repositories and three for Jupyter Notebook as follows: *AI & ML* (applications focused on Data Science) and *Tools* (related to web development, servers, utility scripts, and others) for Python; and *Classification* (Data Science projects aiming to classify data), *Prediction* (projects aiming to predict future data behavior), and *Others* (repositories with study annotations and tools in general) for Jupyter Notebook. Figure 2 presents their distribution per year. Overall, such distribution across domains is similar in both years. Still,

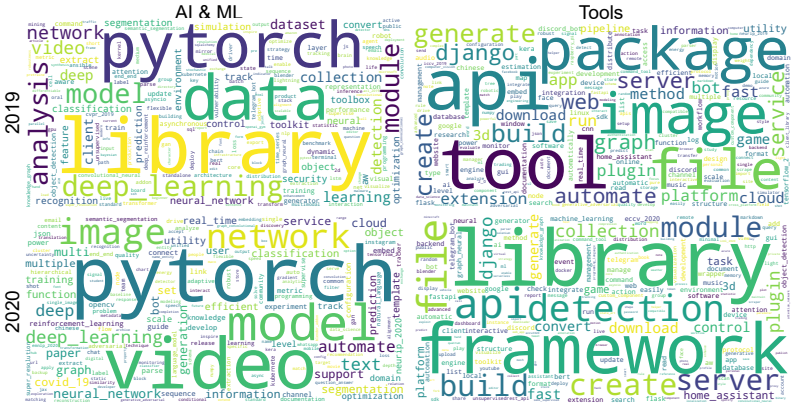


Figure 3. Words in Python repositories descriptions per domain/year

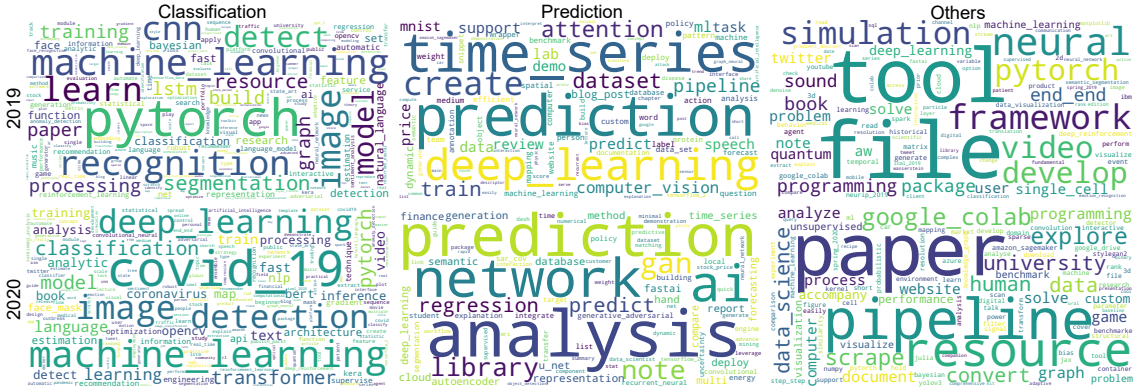


Figure 4. Words in Jupyter Notebook repositories descriptions per domain/year

the gap difference between *Tools* and *AI & ML* has increased for Python, and between *Classification* and the others for Jupyter Notebook, which are better explored next.

Topic analysis. Figure 3 shows the word cloud of the 200 (for easier visualization) most relevant words for Python based on their topic scores for each domain and year. For *AI & ML*, the topics are similar, with some small changes in relevance. For *Tools*, topics are more different between years, but they are still in the context of tools, APIs, servers, and frameworks. Figure 4 shows the word clouds for Jupyter Notebook. The results are generally similar, but in the *Classification* domain, the keyword *covid_19* clearly appears in 2020. Such a result highlights the increased proportion of classification methods due to the pandemic, showing an interest in using Data Science to detect, identify and classify data related to COVID-19. The *covid_19* keyword is the only relevant topic that indicates the repository subject in any domain, as the other words are more generic or technical.

5. Conclusion

This work has three main contributions: comparing different metaheuristics approaches to define the fittest algorithm when tuning LDA for modeling topics from GitHub repositories descriptions; finding domains based on topics from Jupyter Notebook and Python projects; and checking the impact of the COVID-19 pandemic on data science related repositories. Our results found that PSO achieved the highest LDA model coherence in fewer FE than other common metaheuristics. We also identified distinct domains for both

languages: *AI & ML* and *Tools* for Python, and *Classification*, *Prediction*, and *Others* for Jupyter Notebook, which indicate a more specific and applied use in Data Science projects. Our results also suggest an increase in Data Science projects for classifying data, and a growing number of Python repositories unrelated to Data Science. Overall, such initial results highlight the viability of our methodology and raise questions to better understand the pandemics impact. Hence, the next steps include amplifying the study to other languages and repositories' features for correlation analyses, deeper studies over the repositories content in terms of data science methods and datasets, as well as team collaboration, as previous works from our research group [Oliveira et al. 2018].

Acknowledgements. Research funded by CNPq, Brazil.

References

de Oliveira, P. A. M. et al. (2021). Software development during covid-19 pandemic: an analysis of stack overflow and github. In *SEH, co-located with ICSE*.

Gonzalez, D. et al. (2020). The state of the ml-universe: 10 years of artificial intelligence & machine learning software development on github. In *MSR*, page 431–442.

Oliveira, G. P., Batista, N. A., Brandão, M. A., and Moro., M. M. (2018). Utilização de redes heterogêneas para medir a força dos relacionamentos no github. In *SBBD*.

Panichella, A. (2021). A systematic comparison of search-based approaches for lda hyperparameter tuning. *Information and Software Technology*, 130:106411.

Perkel, J. M. (2018). Why jupyter is data scientists' computational notebook of choice. *Nature*, 563:145–146.

Pimentel, J. F., Oliveira, G. P., Silva, M. O., Seufitelli, D. B., and Moro, M. M. (2021). Ciência de dados com reprodutibilidade usando jupyter. In *Jornada de Atualização em Informática 2021*, pages 11–59. SBC.

Ralph, P. et al. (2020). Pandemic programming: How COVID-19 affects software developers and how their organizations can help. *Empir. Softw. Eng.*, 25:4927–4961.

Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. In *WSDM*, pages 399–408.

Saraiva, M. C. and Medeiros, C. B. (2018). Correlating educational documents from different sources through graphs and taxonomies. In *SBBD*, pages 121–132.

Sharma, A. et al. (2017). Cataloging github repositories. In *EASE*, page 314–319.

Silveira, P. et al. (2021). A deep dive into the impact of covid-19 on software development. *IEEE Transactions on Software Engineering*.

Sipio, D. et al. (2020). A multinomial naïve bayesian (mnb) network to automatically recommend topics for github repositories. In *Procs. EASE*, page 71–80.

Tavares, A. C. R., Batista, N. A., and Moro., M. M. (2021). Greed: Github repositories and descriptions. *Zenodo*. DOI 10.5281/zenodo.5138079.

Wang, L. et al. (2020). When the open source community meets covid-19: Characterizing covid-19 themed github repositories. *arXiv*, 2010.12218.