



Modeling Adaptive Data Analysis Pipelines for Crowd-Enhanced Processes

Cinzia Cappiello , Barbara Pernici , and Monica Vitali ^(✉) 

Department of Electronics, Information, and Bioengineering,
Politecnico di Milano, Milan, Italy
{cinzia.cappiello,barbara.pernici,monica.vitali}@polimi.it

Abstract. Information from social media can be leveraged by social scientists to support effective decision making. However, such data sources are often characterised by high volumes and noisy information, therefore data analysis should be always preceded by a data preparation phase. Designing and testing data preparation pipelines requires considering requirements on cost, time, and quality of data extraction. In this work, we aim to propose a methodology for modeling crowd-enhanced data analysis pipelines using a goal-oriented approach, including both automatic and human-related tasks, by suggesting the kind of components to include, their order, and their parameters, while balancing the trade-off between cost, time, and quality of the results.

Keywords: Data analysis pipeline modeling · Crowd-enhanced processes · Human-in-the-loop processes · Goal-oriented process improvement

1 Introduction

Social media analysis can be very effective to support decision processes in citizen science [5,6]. However, the sources of data are not totally reliable and are characterised by high volume and high heterogeneity, also lacking relevant meta-data that might help to identify the items that are actually significant for the decision process. Extracting and selecting relevant data for a specific goal from social media can be very challenging and time consuming. This process can be facilitated and improved by adopting a human-in-the-loop approach, in which the relevance of the data is validated by experts and/or volunteers. Existing approaches combine automatic tools with crowdsourcing and automatic classification techniques to make the data extraction from social media process more efficient and reliable. However, including crowdsourcing in the process generates some challenges both from the cost and time perspectives. In this paper we propose a methodology for improving crowd-enhanced pipelines, modeling their requirements and constraints and the pipeline components and their characteristics. The paper is organized as follows. Related work and research directions in the field are discussed in Sect. 2. Section 3 defines the data preparation

pipeline and its main components. Section 4 provides a model for the adaptive pipeline definition. Section 5 describes the methodology for a goal-oriented pipeline design. An application of the approach is illustrated in Sect. 6.

2 Related Work

Recently, representing and analyzing data science life cycles has received more and more attention. Characterizing machine learning processes, proposing a maturity model and emphasizing the need of assessing the accuracy of the results of a pipeline, is discussed in [1]. The NIST Big Data Reference Architecture [4] proposes a framework in which activities for the big data life cycle management are defined and a system orchestrator enables the execution of different processes. The data science life cycle presented in [12] is analysed focusing on the main steps and their possible infrastructure components, and a customization for specific projects is advocated according to the data sets and the purpose of the data collection. The choices made in the selection of parameters and their motivations are to be considered in the workflow documentation. In [9], several aspects of the data preparation process, defined as “black art” in the paper, are emphasized. In particular a significant issue is the cost of labeling, performed by crowdsourcing. The fact that different data preprocessing pipelines may lead to different analysis results has been also highlighted in [3], where the author proposes the Learn2Clean method to identify the optimal sequence of tasks for the maximization of the quality of the results. In this approach the feasibility of the pipeline in term of cost and execution time has not been considered.

Pipelines for social good projects are analyzed in [11]. In particular the paper discusses the problems of data collection and integration and of managing a large number of annotators in crisis events. In emergency situations, when awareness about the ongoing event is derived also from social media, the data preparation phase is particularly critical as the total workload of emergency operators and involved emergency support online communities is constrained. Some studies analyze the correlation between workload and recall of highest ranked alerts [10]. In [6] the use of social media posts for emergencies has been studied, focusing on extracting images for awareness purposes. In this case, social media data is usually very noisy, with a limited number of relevant information, so automatic filtering must be used for reducing the number of irrelevant posts [2]. In [8] we presented a pipeline in which the preparation phases is augmented with some automatic ML-based filters to remove non-relevant images in a generic way. The present work is motivated by the need of designing such pipelines in a systematic way, considering the specific goals of the collection and analysis pipeline.

3 Components for Adaptive Pipelines

Data analysis always requires the definition of a pipeline in which the different steps show the way in which data are processed throughout their lifecycle.

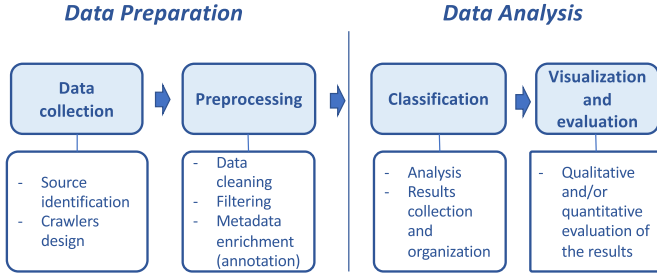


Fig. 1. High level social media pipeline

The high level pipeline is composed of two main phases (Fig. 1): (i) *Data preparation*, related to the definition of the data to collect and to the preprocessing activities needed to transform the data before their analysis and (ii) *Data analysis*, which focuses on the specific analysis (a crowdsourcing task in the considered scenario) to perform and on the visualization and evaluation of the obtained results. Focusing on the preprocessing phase, we aim to build the most appropriate pipeline along the analysis requirements. The pipeline will be built by considering different components. The *preprocessing components* include:

- *Cleaning tasks*: they (i) identify and correct errors and anomalies; (ii) eliminate duplicates; (iii) eliminate inappropriate and useless data.
- *Semantic filters*: they reduce the volume of data to consider on the basis of their value or characteristics. For example, tweets can be selected on the basis of their content (e.g., presence of images) or on the basis of the tweet metadata (e.g., posting time, geolocation data, popularity of the post).
- *Selectors (or Sampling filters)*: they reduce the number of values to analyze. They are characterized by a reduction rate, i.e., the ratio between the output and the input data sets. In our approach, we use the simple random sampling.
- *Metadata enrichment/Annotation tasks*: they add important metadata such as location, topic, image description. Metadata can be simply extracted from the available information (e.g., gather the location of a tweet from the text) or derived from a more complex content analysis, such as a classification of the available images into photos and non-photos, like diagrams or memes.

Most of the preprocessing tasks can be performed automatically or manually, e.g., assigning them on crowdsourcing platforms. Crowdsourcing is also used as a classifier in the data analysis phase since the use of a human-in-the-loop approach can guarantee better results compared to a ML classifier.

There are various components that can be employed to perform preprocessing activities. Selecting a proper combination for a specific data set and a specific goal is not trivial also for expert users and several trials are often needed to find an effective pipeline. Here we focus on this issue by proposing a semi-automatic approach to support data scientists in composing the pipeline for their tasks.

A **Pipeline** \mathcal{P} is a process composed of a set of input sources \mathcal{S} , a set \mathcal{C} of components, which are the elementary work units that collectively achieve the

process objective O_i , within a set of requirements (goals and constraints) \mathcal{R} and a collection of results Res . In this paper we consider only sequential processes.

A **Component** $c_i \in C$ is a self-contained module that performs a specific step of the pipeline. It is characterized by a name, a description, the type of inputs IN_i and outputs OUT_i (e.g., text, image), the fan-in FI_i , i.e., the maximum number of input values allowed, the set of non functional criteria C_i associated with the component (e.g., estimated cost, time, quality), the set IQ_i that describes the impact of the component on the quality criteria described in next section, and the set of configuration parameters $Conf_i$ (e.g., the reduction rate for selectors or the confidence threshold for semantic filters).

4 Adaptive Pipeline Model Definition

In this paper, we propose a methodology for supporting users in the definition of a pipeline and its improvement using a requirement-aware adaptive approach. We introduce a set of criteria to evaluate a pipeline (Sect. 4.1) and we introduce the concept of requirements to express the expected or desired behaviour (Sect. 4.2).

4.1 Evaluation Criteria

We consider three main criteria for assessing the performance of a pipeline and comparing different configurations: (i) cost, (ii) time, and (iii) quality.

Cost. The cost of execution of a pipeline depends on the amount of resources required for processing all the items of the data set. At the component level, the cost can be computed depending on the type of component we are considering:

- the computational cost: in case of automatic components, the cost is related to the computational resources needed for its execution. It depends on the processing capability of the component, expressed by the parameter FI , and on the amount of data items that it has to process;
- the crowd cost: it is the cost of human-in-the-loop components. It depends on the number of items submitted to the crowd. Defining the price per task, that is the amount of money owed to each crowdworker who executes it is a complex and ethical related issue¹ that might affect the quality of the result [7]. Here, we assume that a price, as well as the crowdsourcing platform, has been predefined by the user.

Given the cost of each component, the overall cost of the pipeline $cost_P$ is the sum of the costs of each component according to the number of items that each component has to analyze.

¹ <https://medium.com/ai2-blog/crowdsourcing-pricing-ethics-and-best-practices-8487fd5c9872>.

Time. The time of execution of the pipeline contributes to the efficiency of the process. The time depends on the features of the components as well as on the number of items to analyze. As discussed in Sect. 3, each component is characterised by an estimated average execution time per item. The overall execution time of the pipeline $time_P$ can be computed as the sum of the expected computation time of each component given the expected number of items to process. It is worth noticing that the order in which the components are executed can affect the overall time of execution. This is due to the fact that each component is characterised by a time for processing an item. Also, some components can reduce the size of items in output (reduction rate). Thus, postponing the execution of time expensive components to when the items to be analysed are reduced after several filters can be a strategy to reduce the overall execution time.

Quality. Quality is a criteria that contributes to the effectiveness of the pipeline, measuring the relevance of the items in input of the data analysis phase. More precisely, the quality of the pipeline can be expressed with the following metrics: precision, recall, population completeness, and volume. Considering an input set \mathcal{I} with $m \leq |\mathcal{I}|$ relevant items and an output set \mathcal{I}' with $n \leq |\mathcal{I}'|$ relevant items and $\mathcal{I}' \subset \mathcal{I}$ we can define precision and recall as follows.

Precision p is a metric measuring the ability to select only items that are relevant for the task and is measured as $p = \frac{n}{|\mathcal{I}'|}$. For the filtering components, the average precision at component level is known based on their characteristics, while precision at pipeline level is not known before the pipeline is instantiated. In order to estimate the likely precision of the overall pipeline, we can refer to some precision related metrics at the pipeline level like the maximum, minimum, and average precision of the components of the pipeline.

Recall r is a metric measuring the the ability to keep items that are relevant to the task and is measured as $r = \frac{n}{m}$. As for precision, recall at pipeline level is not known before the pipeline is instantiated. As before, we can refer to some recall related metrics at the pipeline level like the maximum, minimum, and average recall of the components of the pipeline.

Population Completeness d measures the number of data items in the data set belonging to the different classes that the user aims to represent. It ensures the presence of a sufficient number of items for all the classes of interest.

Volume measures the size of the data set in input or output. The volume is the cardinality of the data set $|\mathcal{I}|$. Volume, measured at the pipeline level, ensures that enough data are available for executing the analysis tasks.

4.2 Requirements: Goals and Constraints

When a task is defined, the user defines a set of requirements that the pipeline is expected to achieve, based on the three criteria presented in Sect. 4.1. These criteria are not independent. In fact, high quality negatively affects cost and time. Similarly, reducing time might affect the cost (more resources need to be employed) or quality (same resources are employed thus components have to reduce their precision). Finally, reducing cost might increase the execution time

and/or reduce the quality. According to this, not all the three criteria can be optimised, but a trade-off between them is required. We split requirements in constraints to be satisfied (e.g., maximum cost and time available for computation) and goals to be optimized (e.g., maximize precision and recall) for a pipeline.

The pipeline is associated with a set \mathcal{R} of requirements (see Sect. 3).

A **constraint** $n \in \mathcal{R}$ expresses a desired behaviour measurable through an evaluation criteria. An example of constraint could be “*precision > 0.9*”.

A **goal** $g \in \mathcal{R}$ expresses a desired achievement of the pipeline measurable through an evaluation criteria. An example of goal could be “*minimize(cost)*”. A goal can be associated with a weight expressing the importance of the criteria satisfaction. For the general assessment of a pipeline, we consider the overall optimization function as the weighted sum of all the goals expressed by the user.

5 Methodology for Pipeline Improvement

In this section, we describe the methodology for supporting the user in improving the pipeline. The users willing to extract relevant information from a data source, define a preliminary pipeline based on the task that they are willing to execute. The preliminary pipeline is evaluated according to the criteria. Based on the results, a modification of the pipeline is suggested to the user according to some preliminary knowledge: (i) a set of predefined actions that can be applied to generate a modified version of the pipeline (Sect. 5.1) and (ii) the expected outcome of the applied action on the evaluation criteria (Sect. 5.2). The modified version is also evaluated and a new modification is proposed until a satisfactory solution has been found. In this section we are going to define the actions that can be applied to modify the pipeline and their impact on the evaluation criteria.

5.1 Actions

The pipeline can be modified changing its components or their configuration. Each modification can affect the evaluation criteria, positively or negatively. Here, we introduce two different classes of actions, structural and configuration actions, and their effects on the three identified criteria.

Structural Actions affect the composition of the pipeline. Possible structural actions are: (i) add component; (ii) remove component; (iii) switch components.

The **add component** action AC affects the set of components \mathcal{C} of the pipeline by adding an additional component c . This action potentially affects all the criteria, positively or negatively according to the contribution of the added component. For instance, adding a semantic filter can affect positively the quality if it is able to discard not relevant elements while keeping relevant ones. Having less items, even though the component itself requires time and computational resources to be executed, positively affects the following components of the pipeline reducing their execution time and cost. The **remove component**

action *RC* affects the set of components \mathcal{C} of the pipeline by removing a component c . This action improves the pipeline if the removed component was not relevant or had a limited impact, thus improving quality as well as time and cost. However, removing a relevant component might negatively affect the three criteria. The **switch components** action *SC* affects the order of two components $c_i, c_j \in \mathcal{C}$ of the pipeline. For filtering components, this action has not an effect on the overall quality, since in the end the steps executed will be the same. However, executing a most effective component before a less effective one in filtering out not relevant items might affect the time and the cost of the execution.

Configuration Actions affect the pipeline without changing its structure. They act on the data input (fan-in) or output (fan-out) volume of a single component.

The **increase fan-in** action *FI_I* affects a component c by reducing the reduction rate parameter of the selector preceding it. The effect is positive on quality since less items are discarded (randomly) before the component execution, but it affects time and cost increasing the load for the component. The **decrease fan-in** action *FI_D* increases the reduction rate of the selector. It negatively affects quality while reduces time and/or cost.

For changing the fan-out we can instead change the confidence threshold used by a semantic filter component for classifying relevant and not relevant items. The **increase fan-out** action *FO_I* affects a component c by decreasing its confidence threshold parameter. It improves the recall but decreases the precision, since uncertain items (e.g., false negatives and false positives) are not discarded. It also affects time and cost negatively since a higher volume of items has to be processed by following components. The **decrease fan-out** action *FO_D* increases the confidence threshold parameter of the component. It reduces time and cost, while negatively affecting the recall and improving the precision.

5.2 Requirements and Impact

Each action affects the requirements in some way (Sect. 4.2). As an example, the relationship that we qualitatively discussed in Sect. 5.1 is summarized in Table 1 referring to the semantic filter components. These effects represent a

Table 1. Actions effect on criteria for semantic filters components

Action	Description	Quality	Cost	Time
<i>AC</i>	Add component	– recall/volume/completeness, + precision	+, –	+, –
<i>RC</i>	Remove component	+ recall/volume/completeness, – precision	+, –	+, –
<i>SC</i>	Switch components	No effect	+, –	+, –
<i>FI_I</i>	Increase fan-in	+	–	–
<i>FI_D</i>	Decrease fan-in	–	+	+
<i>FO_I</i>	Increase fan-out	+ recall/volume/completeness, – precision	–	–
<i>FO_D</i>	Decrease fan-out	– recall/volume/completeness, + precision	+	+

general case, however each pipeline will have specific effects according to the components considered and to the task. We define the **impact** $i_{a,e} \in [+,-,n]$ as the effect of an action a on a criteria e . The impact is positive (negative) if the evaluation criteria e improves (gets worse) after the enactment of action a and neutral if no significant change is observed. As it can be observed, for structural actions it is difficult to define an overall effect on cost and time since it depends both on the execution time and the reduction rate of the component. As an example, adding a component implies reducing the number of items to be analysed by the following components. However, the execution time and cost for the component itself have to be considered.

6 Validation

Figure 2 shows how different components have been combined to design a pipeline for extracting visual information from social media. The pipeline execution is supported by the VisualCit tool environment illustrated in [8]. Data preparation phase includes data cleaning, semantic filtering, and enrichment component types (selectors are not shown in the figure). For each type of component, several alternatives can be selected, as listed in the figure.

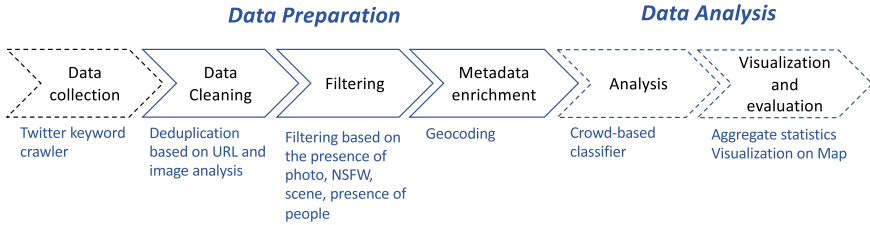


Fig. 2. VisualCit pipeline, revisited from [8]

In this section we briefly discuss two case studies applying the proposed methodology. We start from a general knowledge of the available components and their main characteristics, as shown in Fig. 3. We assume that each component is preceded by a selector in order to control its fan-in.

In the first case study, the goal is to improve an existing data analysis pipeline for providing awareness after a natural disaster. We consider a dataset of images posted in tweets related to the earthquake that stroke Albania on November 26, 2019. The data set consists of 900 images that result from another data analysis pipeline using automatic damage grading classifier components, derived from research at Qatar University [13]. In order to compare human evaluation and automatic classification for damage grading, we used crowdsourcing, in particular 10 annotators, for evaluating the relevance of the images for the analysis task. With the design of an initial data analysis pipeline composed of some filters to select relevant tweets we still noticed a high percentage of non relevant

	person detector	photo detector	NSFW detector	public/priv ate scene	geolocation enrichment	native location
reduction rate	0,21	0,35	0,92	0,80	0,59	0,03
precision	95,81	99,77	99,38	91,81	84,00	100,00
recall	97,15	94,67	98,77	97,26		
time / image (sec)	0,99	0,58	0,33	0,34	10	0,01

Fig. 3. Examples of components parameters

images (80% precision, considering relevance as a parameter for its evaluation), after the filtering provided by the initial data analysis pipeline. Therefore, we apply our methodology to assess if some of the available filtering tools presented in Fig. 3 can improve the quality of the result, retaining a sufficient number of images for the analysis, which is set as $V_{output} > 500$. A combination of NSFW, Photo, Public filters to improve the precision of the result appears a good candidate, considering both the semantics of the filters and their precision and recall characteristics. The application of the filters shows an increase to 90% for precision, while recall decreases by 19%. To further improve the pipeline another candidate component is the geolocation enrichment component since it increases the population completeness. Applying a semantic filter based on location (Albania) after the automatic enrichment with the geolocation enrichment component, only images located in the area of interest (Albania) are considered. However, this component lowers too much the recall (yielding in this case only 200 images) violating the volume constraint needed for the analysis phase.

The second case study concerns extracting visual information on the behavioral impact of COVID-19, described in detail in [8]. Differently from the first case study, this case study is characterized by a very high number of images to be analyzed (approx 1,500,000 per week). Constraints are the computational resources and the size of the crowd, as well as the total time for preprocessing (set to max 12h), with $5,000 < V_{output} < 6,000$ to make the image analysis manageable for a small community crowd (3–10 persons) and to get a number of tweets that would provide some minimal information at country level. A first proposal, and typical approach for a social media pipeline in which post location is needed, is to consider the native geolocation metadata of the tweets to select the items of the initial data set as a first step, and then apply the photo, NSFW, public, persons filter in the order, considering their reduction rate. However, even starting with 1,500,000 tweets, the final estimated volume is 1,516 due to the high selectivity of the native location selector. Time constraints are also not satisfied (15 h). A possible action to modify the pipeline is to remove the native geolocation selector at the beginning of the preprocessing pipeline, which is too restrictive (native geolocation metadata in posts are limited to only 3% of tweets). To provide the needed information, we add an enrichment component for the geolocation. Considering the high computational requirements of the component (see Fig. 3), it should be placed at the end of the preprocessing pipeline. To fulfill the time requirement, we add a selector at the beginning of the pipeline to reduce the number of items to be analysed. With a 1/6 reduction, the

resulting volume is in the desired range. To additionally fulfill the time requirement, we need to assign additional resources to the enrichment component, thus reducing its computational time while increasing costs.

7 Conclusion and Future Work

In this paper we propose a systematic approach to design a data science process, providing an adaptive way to combine different possible alternative components in a pipeline. We addressed this issue by providing a goal-oriented approach for pipeline definition and improvement, proposing a set of evaluation criteria and a set of improvement actions. There is still need of developing this research further in several directions. In particular, there is a need to investigate on methods for classification and characterization of components in specific domains, and to define how the different components can be combined, with respect to the compatibility between them, as proposed in [3].

Acknowledgements. This work was funded by the European Commission H2020 Project Crowd4SDG, #872944.

References

1. Akkiraju, R., et al.: Characterizing machine learning processes: a maturity framework. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 17–31. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_2
2. Barozzi, S., Fernandez-Marquez, J.L., Shankar, A.R., Pernici, B.: Filtering images extracted from social media in the response phase of emergency events. In: Proceedings of ISCRAM (2019)
3. Berti-Équille, L.: Learn2Clean: optimizing the sequence of tasks for web data preparation. In: Proceedings of WWW Conference, pp. 2580–2586. ACM (2019)
4. Chang, W.L., Boyd, D., NBD-PWG NIST big data public working group: NIST big data interoperability framework: volume 6, big data reference architecture [version 2] (2019)
5. Fritz, S., et al.: Citizen science and the united nations sustainable development goals. *Nat. Sustain.* **2**(10), 922–930 (2019)
6. Havas, C., et al.: E2mC: improving emergency management service practice through social media and crowdsourcing analysis in near real time. *Sensors* **17**(12), 2766 (2017)
7. Iren, D., Bilgen, S.: Cost of quality in crowdsourcing. *Hum. Comput.* **1**(2), 283–314 (2014)
8. Negri, V., et al.: Image-based social sensing: combining AI and the crowd to mine policy-adherence indicators from Twitter. In: ICSE, Track Software Engineering in Society, May 2021
9. Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data lifecycle challenges in production machine learning: a survey. *SIGMOD Rec.* **47**(2), 17–28 (2018)
10. Purohit, H., Castillo, C., Imran, M., Pandey, R.: Ranking of social media alerts with workload bounds in emergency operation centers. In: Proceedings of Conference on Web Intelligence (WI), pp. 206–213. IEEE (2018)

11. Scheunemann, C., Naumann, J., Eichler, M., Stowe, K., Gurevych, I.: Data collection and annotation pipeline for social good projects. In: Proceedings of the AAAI Fall 2020 AI for Social Good Symposium (2020)
12. Stodden, V.: The data science life cycle: a disciplined approach to advancing data science as a science. *Commun. ACM* **63**(7), 58–66 (2020)
13. Zahra, K., Imran, M., Ostermann, F.O.: Automatic identification of eyewitness messages on twitter during disasters. *Inf. Process. Manag.* **57**(1), 102107 (2020)