Towards explainable data-to-text generation*

Alessandro Dal Palù[1,3,**], Agostino Dovier[2,3] and Andrea Formisano[2,3]

[1]Università di Parma, Dipartimento di Scienze Matematiche, Fisiche e Informatiche
[2]Università di Udine, Dipartimento di Scienze Matematiche, Informatiche e Fisiche
[3]GNCS-INdAM, Gruppo Nazionale per il Calcolo Scientifico

Abstract

In recent years there has been a renewed burst of interest in systems able to textually summarize data, producing natural language text as a description of input data series. Many of the recently proposed approaches to solve the *data-to-text* task are based on Machine Learning (ML) and ultimately rely on Deep Learning (DL) techniques. This technological choice often prevents the system from enjoying explainability properties.

In this paper we outline our ongoing research and present a framework that is ML/DL free and is conceived to be compliant with *xAI* requirements. In particular we design ASP/Python programs that enable explicit control of the abstraction process, descriptions' accuracy and relevance handling, and amount of synthesis.

We provide a critical analysis of the xAI features that should be implemented and a working proof of concept that addresses crucial aspects in the abstraction of data. In particular we discuss: how to model and output the abstraction accuracy of a concept w.r.t. data; how to identify *what* to say with controlled synthesis level: i.e., the key descriptive elements to be addressed in the data; how to represent abstracted information by means of visual annotation to charts. The main advantages of such approach are a trustworthy and reliable description, a transparent methodology, logically provable output, and measured accuracy that can control natural language modulation of descriptions.

Keywords

Data-to-text, Explainable AI, Answer Set Programming

1. Background

The *data-to-text* generation task consists in automatically generating descriptions from non-linguistic data, such as numeric data or their visual representation in terms of charts, histograms, etc. Over the recent years, there has been a burst of interest in systems able to textually summarize data, such as time-series. For instance, to mention one research line among many, the approach reported in [1] proposes *DataTuner*, neural end-to-end data-to-text generation system. The tool is open source and is supported by the Alexa AI organization.

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy

*Research partially supported by Interdepartmental Project on AI (Strategic Plan UniUD–22-25) and by INdAM-GNCS projects CUP E55F22000270001 and CUP E53C22001930001.

**Corresponding author.

alessandro.dalpalu@unipr.it (A. Dal Palù); agostino.dovier@uniud.it (A. Dovier); andrea.formisano@uniud.it (A. Formisano)

0000-0003-0353-158X (A. Dal Palù); 0000-0003-2052-8593 (A. Dovier); 0000-0002-6755-9314 (A. Formisano)

The relevance of this kind of systems lies in their ability to provide textual descriptions of non-linguistic data in contexts where visual rendering of information represents an obstacle rather than a facilitation to access to knowledge. Compared to traditional forms of presentations, textual description can help, for instance, in making data more accessible to readers/users that are not expert enough to correctly interpret information encoded into a data set. Also, natural language descriptions of data can help readers also in cases where data can be illustrated by means of forms of representations usually considered "self-explanatory", such as charts, histograms, pies, etc. In fact, also in this cases, it might be difficult to read the graphical representation if the amount and/or complexity of data is large.

Further use of text-to-speech tools, can be found in screen readers. In this case the screen reader might exploit textual generator to extract relevant information from the charts contained in a digital document or in a web page. This is of extreme importance in situations where the interpretation of visualizations is made difficult or hindered for people with visual impairments or when human readers have limited cognitive abilities in comprehending and analyzing complex charts.

Many approaches to the data-to-text generation task appeared in the literature and the problem has been been widely studied in various specific domains. For example, [2] and [3] apply a data-to-text generation system to interpret data-records concerning NBA basketball games. In the financial field, [4] propose an approach to the problem of comment generation for time-series of stock prices. In the healthcare domain, [5] present the prototype BT-45 aimed at generating textual summaries of physiological clinical data (for instance, acquired automatically from sensors or entered routinely by the medical staff).

Different solutions have been designed to process specific forms of diagrammatic representations of data. For instance, charts and histograms are the inputs of the approaches proposed by [6] and by [7], while other proposals focus on input data in tabular form [8].

It must be noted that almost all recent systems automating data-to-text processing are based on Machine Learning and ultimately rely on Deep Learning techniques and on Neural Networks training algorithms. This is the case for the DataTuner system described in [1].

There are a few exceptions worth mentioning. In particular, [9] describes a rule-based approach to generate descriptions of inputs from a Dow Jones stock quotes database: the input data are processed by an expert system written in the OPS5 production system language. The BT-45 system [5] implements signal analysis and pattern detection techniques in order to process input time series. The extracted features are then processed and converted into textual summaries by exploiting a corpus of human-authored sample linguistic expression of concepts. Such system is one of the few proposals appeared in the literature not relying on ML/DL techniques. In this sense, it is akin to our approach albeit it is expressly conceived and developed to be used in a very specific application domain. Indeed, it relies on an ontology of medical concepts that must be provided by human experts. Such an ontology is essential both for assessing the importance of patterns in input data and for generating expert-oriented output descriptions.

In this paper we present a framework to solve the data-to-text generation task for input (numerical) time series, which is compliant with explainability requirements. To this aim, we design a system which does not rely on Machine Learning techniques, but leverages Answer Set

Programming to ensure intrinsic explainability and provide explicit control of all main aspects of input analysis and data processing. Section 2 provides an analysis of the xAI features that should be taken into account in developing a tool to solve the data-to-text generation task. In particular, our approach to data abstraction is described in Sections 2.1, 2.2 and 2.3. The output of the abstraction phase consists of a collection of possible alternative descriptions of input data, classified by accuracy and relevance. The selection of the best description(s) is described in Section 2.4, while Section 3 reports on some experimental results. In Section 4 we conclude.

2. The xAI pipeline design

Recent debates about the impact of Artificial Intelligence (AI) on many aspects of everyday life resulted into an increasing awareness about social and ethical implications of the use of AI-based systems. Different regulations and white papers have been developed (see e.g., the 2021 USA National AI Initiative Act and the forthcoming on 2024 EU regulations [10]). EU document introduces a risk-based classification of AI systems based on the principles of ethical AI (cf., [11]).

The term *explainable AI* (xAI, for short) [12, 13] identifies a broad set of features that a high-risk system should implement: transparency (or glass box approach), ethics, the capability to support results in terms of intelligible descriptions, accountability, security, privacy, and fairness. A system built on top of such prerequisites can be trusted and embedded into a panel for high-stake decisions, given its capability of describing at high-level the justification of the inner deductions that brought to a particular answer.

The machinery used by most Machine Learning (ML) and in particular by Deep Learning (DL) hides high-level (symbolic) interpretation about the learned parameters (black box approach). Recent efforts try to recover explainability by observing the trained networks (post-hoc explanations [12]), e.g., textual captioning and/or visual explanation by examples. However, there are concerns about the reliability of these approaches, if still driven by black box approaches [14].

We prefer designing an intrinsically explainable approach. We selected a Logic Programming framework, where knowledge can be expressed, rather than learned and hide, and results derive from the application of resolution techniques that guarantee the soundness of the resulting interpretation. We select Answer Set Programming (ASP) and Clingo [15], given its flexibility in modelling knowledge and efficiency in reasoning. Another advantage is the fast prototyping and the ease in identifying conceptual errors in the model during the developing stage.

In this work we focus on time series, where a quantity is measured and sampled n times at certain frequency along the time domain. We assume that time (on x-axis) and scalar measure (on y-axis) are both normalized into the $0..1$ interval. Our approach is general and specific knowledge about time evolution of data can be added in a compositional fashion (e.g. electrocardiograhpy data provide peculiar shapes, meteorological data). As a proof of concept we describe the data in terms of a combination of basic descriptors that originate from linear shapes (increase, decrease and constant), peaks/valleys and steps (constant-increase-constant pattern). Multiple descriptions of the same portion of the series can coexist, since different shapes can be observed. Shapes should be simple, defined by a few parameters and associated to simple descriptions in natural language.

The pipeline is composed of three steps. The first step is the abstraction of data, where raw data are processed to detect basic local properties. The second step builds a set of abstractions that combine local properties into descriptions that cover larger regions the domain. A third step selects a combination of descriptions (*what* to say), that provide enough reliable information about the data. Finally, each description is translated into natural language (*how* to say) according to its accuracy and relevance computed during the process. In the paper we omit actual text string generation, since output contain enough information to construct a logical narration structure (thanks to the time and inclusion relationships among descriptions) and to modulate qualitative adverbs (from accuracy and relevance information). For effective and intuitive feedback we show equivalent graphical annotations over original data.

2.1. Accuracy and relevance

The most crucial aspect in the semantic interpretation and abstraction of a general data series is the need to measure the faithfulness of the abstraction process. The system should provide, as self evaluation of the process, the amount of confidence about each description. This type of explainability pairs the white box approach, where the process itself can be investigated (by looking at the logic program that models the process itself).

The bottom up approach in the abstraction of features from data to high-level description starts from numerical fitting of simple shapes over the data. Such action, described below, is the link between raw data and logic handling of information. Fitting itself carries an error (typically Root Mean Squared Error, RMSE) and this is the first measure of *accuracy* that propagates along the process. If data provide a low accuracy in fitting a particular shape, the high level description is necessarily weak.

The logic program that merges low-level shapes into more complex ones (e.g., an increasing sequence followed by a decreasing one may form a peak) needs to account for another type of errors. The combination of descriptions into an higher-level description must measure the deviation from the ideal model. For example, an ideal peak must contain an increasing trend followed by a decreasing one. However, a soft preference can penalize accuracy when the two halves are not symmetric. Therefore, accuracy of a description combines fitting errors and abstraction errors.

Any description with associated high accuracy can be safely selected, since both visually and conceptually there is a high correlation with data.

Handling at high level, through a logic program, the computation of descriptions, rather than having a large set of candidate shapes to be fitted on the dataset, provides greater flexibility and avoids a relevant computation burden.

The *relevance* of each description is an orthogonal property that captures the importance of the shape, in terms of extension in the measure domain. For most shapes it can be seen as the coefficient that scales the y-axis. The combination of accuracy and relevance are at the basis of the choices for descriptions that cover the series. Moreover, it provides the sufficient ingredients for the generation of meaningful natural language characterizations. For example, accuracy can modulate the confidence expressed in the sentence about a particular shape, while relevance can control qualitative adverbs.

2.2. First step: data abstraction

The first step has the goal of retrieving simple features that describe the time series. Such features become the basic building blocks for further abstractions handled by ASP program in later steps. In our examples, we use polyline fitting to describe the original data with a controlled RMSE. Other types of fitting can be used (e.g., specific patterns can be convoluted over the series as well as other parametric curves).

We take advantage of Python's library `scipy.interpolate` and its function `UnivariateSpline` that is able to efficiently fit general splines. We selected the spline degree equal to 1, in order to produce a polyline made of simple segments. Depending on the selected fitting accuracy (provided as input), an adequate number of segments are placed over the curve. As an example, Figure 1 depicts a series resulting from the search term "Concert" in the last 5 years (from Google Trends). We select 9 *levels* of increasing fitting accuracy for polylines. Throughout the paper we refer to a polyline's level as the i-th version of fitting obtained with a decreasing sequence of RMSEs. Blue lines represent the original data, while red segments are the result of the fitting. Higher accuracy requires a larger number of segments, while describing finest details. On the other side, setting a high error, allows to capture coarser and general trends. In Figure 1, the top row approximates a rather constant behaviour that mutates to a large valley in the central row. The bottom row captures tiny spokes and almost noisy behaviour.

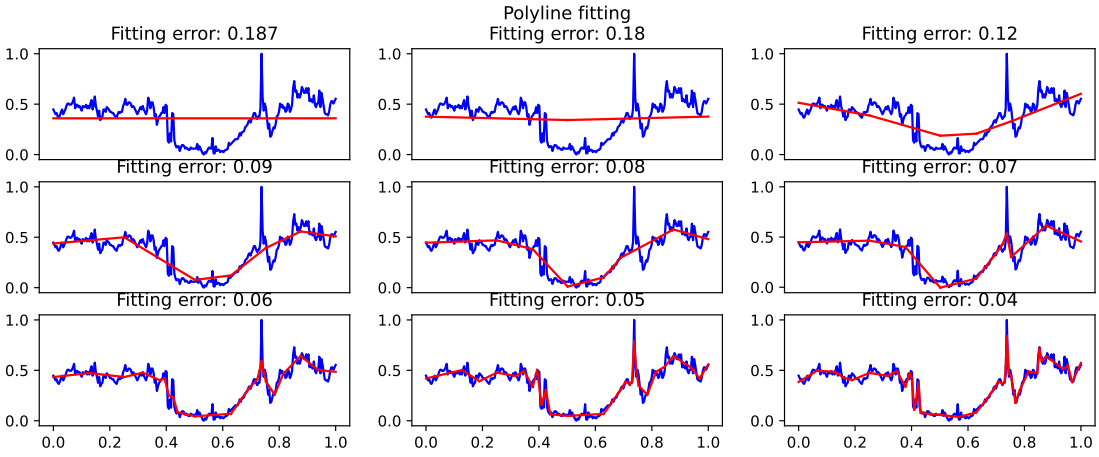

Figure 1: Example of fittings with decreasing RMSEs

A polyline is associated to a global RMSE, but single segments of the polyline can reveal different local RMSEs. In Figure 2 we depict, from higher fitting error to lowest, the measure of local RMSE in a black-red-yellow palette corresponding to polylines of Figure 1. It can be noted that the initial large error for the central valley is reduced while global RMSE gets reduced. The error caused by the sharp peak on the right requires a larger number of segments in order to be significantly reduced. In our examples we defined $n/2$ (at most 128) bins over the time domain to measure local errors. This allows to compare segments' properties even among different polylines, since they are not necessarily aligned on x coordinates.

Fitting errors should be taken into account when producing a shape that describes the data.

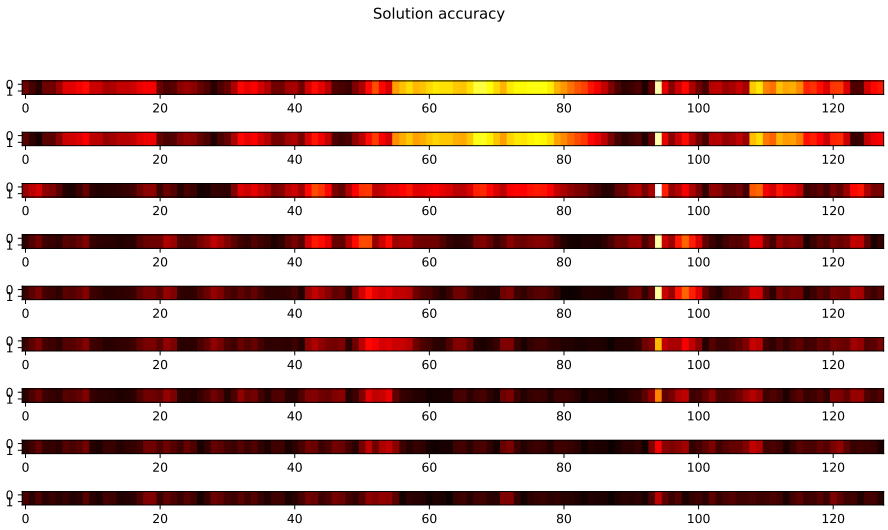

Figure 2: Local accuracy for each polyline fitting

For example, claiming that the series of Figure 1 has an overall constant behaviour implicitly carries a relevant error in the central part of the domain (yellow area in the top row if Figure 2). Depending on the domain knowledge, the amount of error can be either satisfactory or not in relation to the desired description approximation. The pipeline should provide the means to select the minimal degree of acceptability for any description. The simplest way to proceed is to expose RMSE information to next steps and allow user to set a parameter for minimal acceptable error.

The output of this processing prepares a set of ASP facts that compose the basic knowledge about the series, in terms of: lists of segments classified by their polyline level (`desc(ID,Level)`); segment's properties (`desc_info(ID, Type, Value)`, details below); local RMSE for each bin covered by a polyline level (`desc_accuracy(Level,Bin,RMSE)`, equivalent to the content of Figure 2) are output. Details about a segment include x and y coordinates of ending points, minimal and maximal y encountered in original data and gradient. Since ASP deals with integral quantities, all values are scaled to 0..100 range.

2.3. Second step: bottom up abstraction hierarchy

Let us focus on a specific polyline ℓ (level) out of the set generated in previous step. A sequence of segments `desc(`i`,`ℓ`)` represents the finer knowledge about polyline. The goal is to merge adjacent sequences of segments and produce potentially relevant abstractions that capture particular properties.

We define merge rules depending on the type of property that is investigated. We currently have three categories: monotone behaviour, second order zero (peak and/or valley) and step function (low - high - low sequence of first order derivative magnitude). Other patterns can be

derived similarly, by building on top of such structure (e.g. periodicity), while other shapes may require different raw data pattern matching.

Monotone arrangements can be directly deduced from `desc_info(ID, Type, Value)` facts, and an *increasing* (*decreasing*) subcategory is selected if the angular coefficient of the segment is positive (negative). Accuracy of such descriptions only depends on RMSE, since these are basic shapes. In particular (recall that we scale numbers in the range 0..100), we assign an accuracy `Acc` of $100-RMSE$, meaning that 100 is a segment that matches the original data, while lower values show noisy fitting. Relevance `Rel` of the segment is associated to the difference of y values between extremes. Intuitively it represents the impact of the change in the plot. These basic abstractions are stored as facts `abstract(A,B,L,monotone,SubC,Acc,Rel).`, where A..B is the range of segments indexes involved (in this specific case $A = B$, since we abstract a single segment), `L` is the level ℓ that selects the polyline that contains the segment, `SubC` is the subcategory.

Let us discuss now the merge rules for consecutive abstracted objects. The general bottom up procedure joins consecutive abstractions and produce new ones with larger segments span. The new abstraction is classified by a category and subcategory, as well as accuracy and relevance, depending on rules that model the abstraction process. We therefore produce a set of horn rules that models each abstraction type.

In Table 1 we report some examples of simple combination rules that drive the bottom up process. The output column is the result of the concatenation of 2/3 lower level abstractions. Bottom elements are `abstract/7` facts described above. Let us show the actual rule for the first case listed in the table (simple merge of two consecutive increasing regions). In Figure 3 we show a rule for producing a new abstraction for the range $A..C$ at level L of subcategory increasing. The computation of the new accuracy `Acc` inherits the accuracy of the two components. We found that selecting the minimal one among the two is rather plausible. For fitting errors this is certainly true (remember that accuracy is computed as 100-RMSE and minimal accuracy is equivalent to maximal error), while for the error added by the merge operation itself, we believe that we can inherit the weakest (lower) component's relevance. Obviously, any different measure for degradation caused by merging operators can be easily introduced. Relevance `Rel` is computed by means of support predicates `structure` that merge `desc_info` facts by joining any possible concatenation of sub-regions and that show structural properties of the segments (in this case the y value for first and last point of the sequence of segments).

Let us discuss a slightly more complicated case that captures the subtle impact on accuracy when merging incompatible segments. For example, we model a region (made of segments $A..B$) defined as increasing with accuracy `Acc1` and relevance `Rel1` followed by a region (made of segments $B+1..C$) defined as decreasing with accuracy `Acc2` and relevance `Rel2`. We can abstract the union of the regions as decreasing under certain conditions. If the increasing region is particularly small, both in x or y axis, in comparison to the second region, then it is plausible to consider it as a detail that can be smoothed out in a more general abstraction. Nevertheless, we account for such abstraction error, which is not related to the fitting accuracy of the base segments, but rather on the intuitive semantic error due to the merge. Figure 4 details the computation of a Penalty that is subtracted from accuracy of the decreasing region. The Penalty depends on the ratio of the two regions: if increasing part gets irrelevant, the penalty becomes null. If the increasing part covers half of the decreasing part, the penalty becomes 50. When the

Output		Segment 1		Segment 2		Segment 3	
Category	Subcat	Category	Subcat	Category	Subcat	Category	Subcat
Range [A..C]		Range [A..B]		Range [B+1..C]			
monotone	incr	monotone	incr	monotone	incr		
monotone	decr	monotone	decr	monotone	decr		
peak/valley	peak	monotone	incr	monotone	decr		
peak/valley	valley	monotone	decr	monotone	incr		
Range [A..D]		Range [A..B]		Range [B+1..C]		Range [C+1..D]	
step	incr	monotone	const	monotone	incr	monotone	const
step	decr	monotone	const	monotone	decr	monotone	const

Table 1
Basic abstraction rules for merging categories

```
abstract(A,C,L, monotone, SubC, Acc, Rel):-
                SubC=increasing,
                abstract(A,B,L, monotone, SubC, Acc1, Rel1),
                abstract(B+1,C,L, monotone, SubC, Acc2, Rel2),
                Acc=#min{Acc1;Acc2},
                structure(A,B, L, firsty, Fy1),
                structure(B+1,C, L, lasty, Ly2),
                Rel=Ly2-Fy1.
```

Figure 3: ASP code for merging two increasing abstractions

```
penalty(2,50;3,40;4,30;5,20;6,10;7,5;8,0;9,0;10,0).
abstract(A,C,L, monotone, decreasing, Acc2 - Penalty, Rel):-
                    abstract(A,  B,L, monotone, increasing, Acc1, Rel1),
                    abstract(B+1,C,L, monotone, decreasing, Acc2, Rel2),
                    Dx1=Maxx1-Minx1+1, Dx2=Maxx2-Minx2+1,
                    Dy1=Maxy1-Miny1+1, Dy2=Maxy2-Miny2+1,
                    Dx2 / Dx1 = Times1, Dy2 / Dy1 = Times2,
                    Times=#min{Times1;Times2},
                    penalty(Times,Penalty),
                    Rel = |Rel1-Rel2|.
```

Figure 4: Merge of incompatible categories

two regions become comparable (Times ratio is equal to 1) the body fails, since the two regions would merge in something that has no meaning from monotone category point of view. In this case, relevance gets reduced since the maximal coverage on y axis depends on the difference between the two regions extensions (or equivalently the difference between the y of the peak and the highest between the first and last y values).

Such kinds of rules allow to produce multiple categories with multiple subcategories abstractions for the same range. Some differences in accuracy and relevance may arise. Moreover, even

the same category and sub category could show different facts with different combinations of accuracy,relevance since there is no associativity of operators in the merge rules. In order to simplify a potential explosion of alternative (and no so interesting) options, the ASP program selects, for each range, category and subcategory, the abstraction with maximal accuracy and relevance found. The rationale is that among them, there is a combination of merges that is able to support a maximal accuracy and relevance.

Let us mention the peak generation, where the accuracy estimation receives another type of penalization. In this case, we prefer an ideal shape that presents symmetric increasing and decreasing ramps (both along x and y axis). Similarly to the previous case, we penalize merges of unbalanced parts. Relevance in this case depends on the minimal one among the two regions.

We also want to model a monotone constant model. This is complementary to a increasing/decreasing behavior. In this case, we inherit the same accuracy, while relevance gets high if the increasing/decreasing relevance is low. Moreover low relevance for any other abstraction (valleys, peaks and steps) can produce high relevance constant descriptions.

The bottom up production of abstractions is handled by Clingo resolution of the ASP program and requires a simple stratified T_P resolution with a single stable model.

2.4. Third step: selection of descriptions

Given a set of polyline fittings (recall Figure 1), we index them by *levels* of improved fitting RMSE. Each level is associated to a polyline and related abstractions for various ranges of segments, classified by accuracy and relevance. This dataset is at the basis for the next task: the selection of important descriptions that arise, at any level.

An generic description of data requires a multi-scale approach: a comprehensive observation should cover both large scale properties (captured by numerically low levels, higher RMSE polylines) and small scale properties (numerically high level, lower RMSE polylines). The correct levels at which coarse trends (*summary* descriptions) and fine grained trends (*detailed* description) appear depend on the series. Moreover, the same detail can be detected at many levels, with different amount of accuracy and relevance. For example, referring again to Figure 1, we list the same valley property detected at different levels. For top row, starting from left, no valleys are found (only a constant description with accuracy 69/100 and relevance 100/100); for the central fitting a faint valley is found, covering the whole dataset, with accuracy of 64/100 and relevance 3/100 (very weak, barely noticeable); the right figure produces a valley description again over the whole dataset with 68/100 of accuracy, but 33/100 of relevance. On the central row, the left figure is able to produce a valley description of 72/100 accuracy and 42/100 relevance limited between 0.25 and 0.87 along x axis, which becomes rather usable as actual description. The challenge is to select non redundant details among potentially overlapping descriptions from different levels of description.

We define a logic model that promotes the creation of a fluent and effective narration of descriptions. In our view an intuitive, yet not exhaustive, set of goals is: to cover both summary and detail descriptions; avoid to describe redundant descriptions (same category over same region, also from different levels); control the amount of synthesis by means of a limited number of description to be used; to cover each point of the series by at least one description; to maximize the quality of the descriptions (minimal rmse = best accuracy and maximal relevance).

```
cost(P,Cost,D) :- position(P),
    talk_about_top(D),
    Cost = #max{Rel1,D1:
                talk_about(D1), includes_trans(D,D1),
                output_ok(D1,Start1,End1,_,_,_,_,Rel1),
                desc_info(Start1, minx, S1), desc_info(End1, maxx, E1),
                S1<=P,P<=E1;
                0,dummy}.
```

Figure 5: Cost definition

Our efforts to convert these guidelines into an ASP program provide the advantage of being very flexible about accommodating additional/modified goals. The possibility to interact with a logic program, implements the xAI guideline for a white box approach. Even if our model contains an optimization, compared to pure constrained based solutions, we appreciate ASP's flexibility in mixing constraints and aggregates into simple predicates that define the search space.

Let us present our model step by step. We start by defining two minimal thresholds for selecting descriptions that are worth considering: `min_accuracy(60)` and `min_relevance(2)`. This cuts our descriptions too inaccurate and/or invisible. We define the facts that are filtered as follows: `description(ID,Start,End,L,Cat,SubC,Acc,Rel)`. Basically, the variables have been discussed above: `Start` and `End` refer to the segment number of the polyline at level `L`; `Cat` and `SubC` identify the category and subcategory; `Acc` and `Rel` describe accuracy and relevance. Each `description/8` is a potential candidate to be talked about. We therefore define decisional variables, by means of the `talk_about/1` predicate, as follows:

$$0\{\texttt{talk\_about(D)}\}1 :- \texttt{description(D,\_,\_,\_,\_,\_,\_,\_)}.$$

We conclude with the definition of the cost function. We define a `cost/3` predicate that determines the cost for each description D and for each position P. Figure 5 describes the ASP code. In order to compare segments that may cover unpredictable and different regions among different levels, we map description properties to the bins introduced for RMSE evaluation. Each description votes its Relevance to each position P. If the position is not covered, the default is set to 0. In case multiple included descriptions are present, the best relevance reaches is retained. Summary and detail optimization will make different usage of `cost/3` predicate.

2.4.1. Summary

The optimization cost for summary retrieval considers for each position the best cost provided by any description at that position:

```
best_cost_position(C,P) :- position(P),
    C=#max{Cost:cost(P,Cost,_); 0}.
    #maximize{Cost,P: best_cost_position(Cost,P)}.
```

```
includes(D1,D2):-
    talk_about(D1),talk_about(D2),D1!=D2,
    description(D1,Start1,End1,L1,_,_,Acc1,Rel1),
    description(D2,Start2,End2,L2,_,_,Acc2,Rel2),
    L1=L2,
    desc_info(Start1, minx, S1), desc_info(End1, maxx, E1),
    desc_info(Start2, minx, S2), desc_info(End2, maxx, E2),
    S1<=S2, E2<=E1,          %%% D2 is contained in D1
    (E2-S2+1)*3 > E1-S1+1.   %%% at least 1/3 covered
```

Figure 6: Example of descriptions inclusion

This allows to consider potential overlaps and forces to cover the largest portion of the series. Assume that `talk_about_top(D)` is equal to `talk_about(D)` and that `includes_trans(D,D1):- D1=D, talk_about(D)`. Practically, we account for any descriptions regardless their inclusion relationships.

Finally, we set a small maximal number (`max_desc(M)`) of descriptions that form the summary: $:-\texttt{max\_desc}(\texttt{M}), (\texttt{M}+1)\{\texttt{talk\_about}(\texttt{D})\}$.

We expect that the optimization selects large and most relevant details, probably from lower levels of fitting. Even if details provide on average higher relevance, their span is limited and therefore discouraged by the optimization function.

2.4.2. Details

The model for retrieving details builds on the resolution and optimization of Summary model, that produces the set of descriptions named `talk_about_summary(D)`.

The summary descriptions are forced into the interpretation, so that the search can select additional descriptions. However, this search takes care about potential inclusions in terms of space and in terms of semantic abstraction. This avoids to talk about the same concept multiple times. Basically, fresh details about an uncovered description are searched.

The relation of inclusion between pairs of descriptions `includes(D1,D2)` captures cases of redundancy. This is useful to better evaluate a description accuracy/relevance also in terms of sub-regions that provide better estimates. This relation partitions the set of descriptions and identifies representatives that are not included by other descriptions. In Figure 6 we describe an inclusion relation between two descriptions belonging on the same level. Basically we check that the range over x axis of D1 contains D2 and that it is no larger than 3 times than D2 (this includes similar versions of the same descriptions).

We also consider inclusions among levels. In this case we require L1<L2 and that they intersect for at least on third of their union. However, we restrict the categories to be the same or valley/peak/steps to include monotone descriptions. In this way we skip simpler features that contribute in building more complex descriptions.

Since includes relation forms a partial order, we select the top elements in the lattice (predicate

`talk_about_top/1`), being the ones that are most extended in space and/or informative:

```
included(D2) :- includes(D1,D2).
talk_about_top(D) :- talk_about(D), not included(D).
```

The cost function for details selects the maximal relevance for a top description and maximizes the sum. Note that included descriptions contribute to the improvement of `cost/3` but their cardinality is irrelevant with respect to the function.

```
best_cost(C,D):-talk_about_top(D),
          C=#max{Cost:cost(_,Cost,D)}.
#maximize{Cost,D: best_cost(Cost,D)}.
```

We also require that details do not contain monotone category, since it is too simple to serve the purpose. Again we set the maximal number of details to be included in the solution.

This optimization favours most prominent descriptions, regardless their span among time domain. Therefore is suitable for capturing tiny but relevant descriptions.

2.5. Towards natural language

Description relationships among time allow to establish a narrative order. Summary and details descriptions prepare two level descriptions. Accuracy and relevance can be easily translated to correct adverbs ranges. In summary, it is conceivable that such information can be translated into natural language with little further processing.

For the purpose of validating the results, we provide a visual annotation that comes directly from descriptions and represents an additional value of the approach.

3. Results

In implementing our tool, we took advantage of Potassco Clingo Python's API. The process is handled by a Python program that controls the pre-computation of input, the handling of ASP programs execution and to converts outputs to text and enriched plots. Results are plotted with Matplotlib and enriching visual description are computed directly from the description structure output by the program.

In Figure 8 we show four time series and results for the request of 2 summary descriptions and 4 details. The green-red bar at bottom shows the pure RMSE accuracy provided by the overlap of selected descriptions. The first one is a synthetic periodic function: the overall behaviour is correctly captured, even if a short constant region is selected. Probably a small weight / cutoff on extension could solve the problem. Even if periodic shape is not defined, peaks and valley begin to appear. We expect that additional extensions can capture periodicity and amplitude relationships. The second series (keyword Udine on Google Trends over last 5 years) shows two constants summaries: the largest one is rather relevant, even if slightly decreasing. Accuracy is around 80/100, given the noisy dataset. Details chosen are among the most relevant ones. The third series (keyword Concert) shows a single constant behaviour even if an accuracy of only 69/100 is reached. Details compensate it by finding a large valley, surrounded on the left

(descending red arrow) by a step like function (again with 66/100 accuracy). Main peaks and valleys are found. Finally last series (keyword Lockdown) shows a nice summary, while details gets into the almost noisy behaviour.

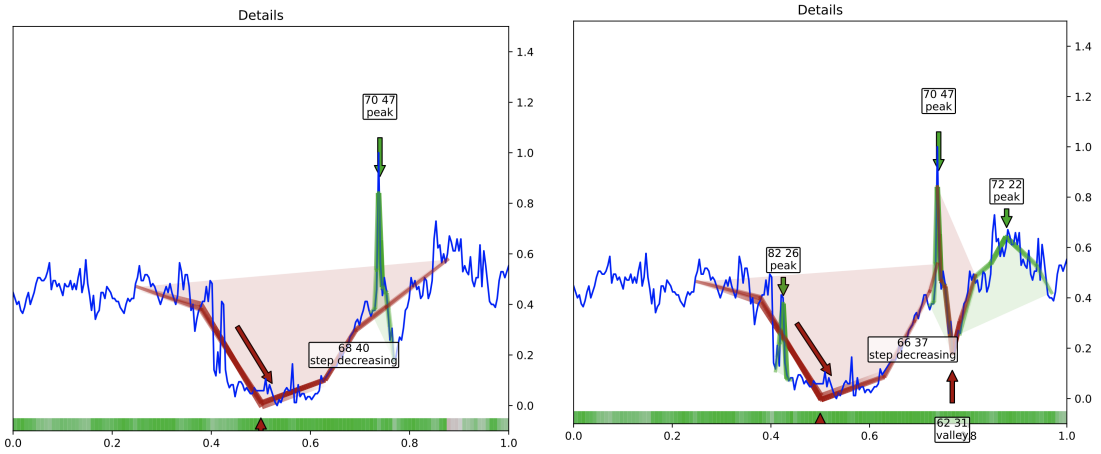

Figure 7: Comparison of 3 and 6 details request

We now report on the synthesis and detail selection capabilities of our model. We select, for 1 summary description, details equal to 3 and 6. Figure 7 shows that with three descriptions, the valley is extended far more on the right, while the introduction of more details allows to cover almost any area of the series without fine details related to noise.

Computational times for ASP programs are around seconds on a 16 threads launch on a 2,3 GHz Intel Core i9 8 core CPU. Grounding is not an issue for the model described.

4. Conclusions

We presented a general framework that addresses the problem of automatic interpretation of time series. We design a novel xAI pipeline, based on ASP, that is transparent and provides feedback about the abstraction errors of the process. We also show how to control synthesis and the definition of summary/detail descriptions.

The model in intrinsically modular and it can be expanded rather easily. We provided some examples of visual annotation to validate the program. Natural language generation is a future work that relies on the already processed features about accuracy and relevance of the descriptions.

In the future we plan to extend the description categories, to introduce comparisons about multiple comparable series over the same time span, to correlate multi-dimensional features along time and to apply this technology to high-stakes domains and critical systems monitoring (medical data, governance data in the Academia).

References

[1] H. Harkous, I. Groves, A. Saffari, Have your text and use it too! End-to-end neural data-to-text generation with semantic fidelity, in: D. Scott, N. Bel, C. Zong (Eds.), Proc. of COLING'20, Int. Committee on Computational Linguistics, 2020, pp. 2410–2424.

[2] R. Puduppully, L. Dong, M. Lapata, Data-to-text generation with content selection and planning, in: Proc. of AAAI'19, IAAI'19, EAAI'19, AAAI Press, 2019, pp. 6908–6915.

[3] S. Wiseman, S. M. Shieber, A. M. Rush, Challenges in data-to-document generation, in: M. Palmer, R. Hwa, S. Riedel (Eds.), Proc. of EMNLP, ACL, 2017, pp. 2253–2263.

[4] S. Murakami, et al., Learning to generate market comments from stock prices, in: R. Barzilay, M. Kan (Eds.), Proc. of ACL'17, Volume 1, ACL, 2017, pp. 1374–1384.

[5] F. Portet, et al., Automatic generation of textual summaries from neonatal intensive care data, Artif. Intell. 173 (2009) 789–816.

[6] J. Obeid, E. Hoque, Chart-to-Text: Generating natural language descriptions for charts by adapting the transformer model, in: B. Davis, Y. Graham, J. D. Kelleher, Y. Sripada (Eds.), Proc. of INLG'20, ACL, 2020, pp. 138–147.

[7] A. Balaji, T. Ramanathan, V. Sonathi, Chart-Text: A fully automated chart image descriptor, CoRR abs/1812.10636 (2018).

[8] T. Liu, K. Wang, L. Sha, B. Chang, Z. Sui, Table-to-text generation by structure-aware seq2seq learning, in: S. A. McIlraith, K. Q. Weinberger (Eds.), Proc. of AAAI'18, IAAI'18, EAAI'18, AAAI Press, 2018, pp. 4881–4888.

[9] K. Kukich, Design of a knowledge-based report generator, in: M. P. Marcus (Ed.), Proc. of ACL'83, ACL, 1983, pp. 145–150.

[10] European Commission, Proposal for a regulation laying down harmonised rules on artificial intelligence, 2021. https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence.

[11] J. Meszaros, J. Minari, I. Huys, The future regulation of artificial intelligence systems in healthcare services and medical research in the European union, Frontiers in Genetics 13 (2022). URL: https://www.frontiersin.org/articles/10.3389/fgene.2022.927721. doi:10.3389/fgene.2022.927721.

[12] A. Barredo Arrieta, et al., Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Inf. Fusion 58 (2020) 82–115.

[13] A. Adadi, M. Berrada, Peeking inside the black-box: A survey on explainable artificial intelligence (XAI), IEEE Access 6 (2018) 52138–52160. doi:10.1109/ACCESS.2018.2870052.

[14] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019) 206–215.

[15] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012.

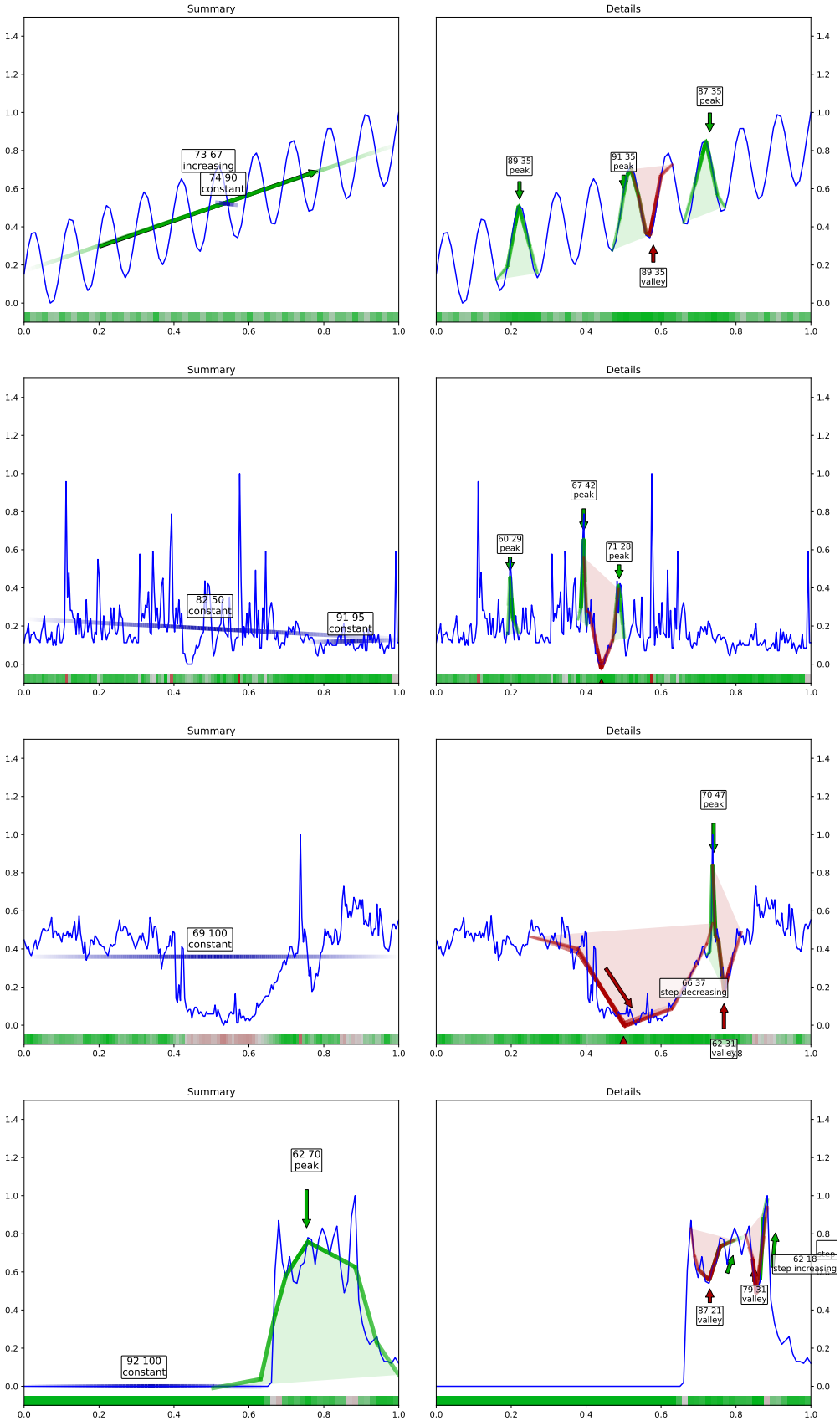

Figure 8: Visual annotations for some series (accuracy and relevance are reported in text boxes)