Language Driven Analytics for Failure Pattern Feedforward and Feedback

Min Jian Yang, Yueling (Jenny) Zeng, Li-C. Wang
University of California, Santa Barbara
Santa Barbara, California 93106

***Abstract*—In the context of analyzing wafer maps, we present a novel approach to enable analytics to be driven by user queries. The analytic context includes two aspects: (1) grouping wafer maps based on their failure patterns and (2) for a failure pattern found at wafer probe, checking to see whether there is a correlation to the result from the final test (feedforward) and to the result from the E-test (feedback). We introduce language driven analytics and show how a formal language model in the backend can enable natural language queries in the frontend. The approach is applied to analyze test data from a recent product line, with interesting findings highlighted to explain the approach and its use.**

1. Introduction

Wafer Map Pattern Recognition (WMPR) is a problem that has been studied in the field of semiconductor manufacturing for decades. A notable example is the dataset called WM-811K reported in [1]. Many works were published based on the dataset, including those employing a feature-based learning approach [1][2][3][4] and those employing a deep learning approach [5][6][7][8][9][10]. Wafer maps in the WM-811K are classified based on nine pre-defined pattern classes. Given such a dataset, it is intuitive to treat WMPR as a multi-class classification problem.

One of the classes defined in WM-811K is called "Edge-Local" indicating that the pattern is "local" (in contrast to a full ring) and also along the wafer edge. Suppose we follow this pre-defined class. Using three months of test data (Period 1,2,3) collected from a product line, Figure 1 shows an application context considered in this work.

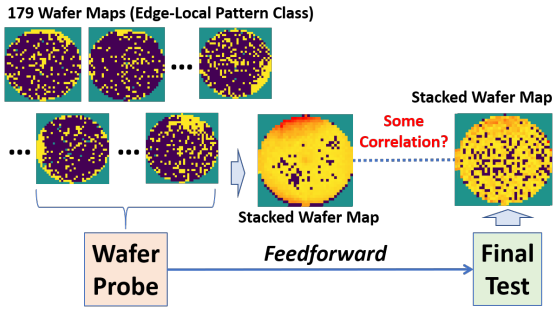


Figure 1. The feedforward context considered in this work

Suppose we have a way to extract a group of wafer maps all containing an "Edge-Local" pattern. Figure 1 shows that there are 179 such wafer maps found from the wafer probe test data. In our application context, knowing this information is not enough. In addition, we would like to see if there is a "correlation" between the failure pattern and the final test result. To do so, we generate two plots: a *stacked wafer map* of the 179 wafers based on their fails from the wafer probe, and a stacked wafer map based on their fails from the final test. Figure 1 shows these two stacked wafer maps. On each map, the color gradient is set from yellow to red where yellow means 1 fail and red is set to the max number of fails across all die locations.

From the wafer probe's stacked wafer map, we see a concentration of fails along the wafer edge from roughly 10 o'clock to 1 o'clock. Suppose we use the phrase "fails spread from 10 o'clock to 1 o'clock along edge" to describe it and give it a pattern name: "Edge-10-to-1". On the final test's stacked wafer map, it seems that there are also more fails in the proximity. The pattern is not apparent though.

The proximity relationship between the wafer probe fails and the final test fails might be considered as a form of correlation. Form the result presented in Figure 1, we would like to conduct further analysis to see if we can find another pair of stacked wafer maps where the correlation of the two maps is more isolated (on fewer wafers).

1.1. Refined analysis

To refine the analytic result, we narrow down the selection of the wafer group for plotting the stacked wafer maps. In Figure 1, the group has 179 wafers. By focusing on the new pattern class "Edge-10-to-1", we could narrow down to 80 wafers maps exhibiting the pattern. Among these 80 maps, the pattern is more prominent on 59, while the rest 21 can be argued to contain another pattern. Figure 2 shows the two cases, based on the 80 maps and the 59 maps.

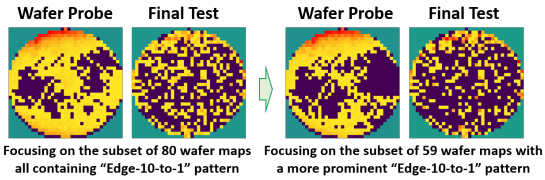


Figure 2. Refined analyses by narrowing down the wafer group selection

As we narrow down the wafer group to 80 wafers and then, to 59 wafers, the correlation is more isolated on fewer wafers. Following this thinking, we further split the 59 wafers by their month labels. This results in two groups: the Period-1 with 53 wafers and the Period-3 with 6 wafers. Their stacked wafer maps are shown in Figure 3.

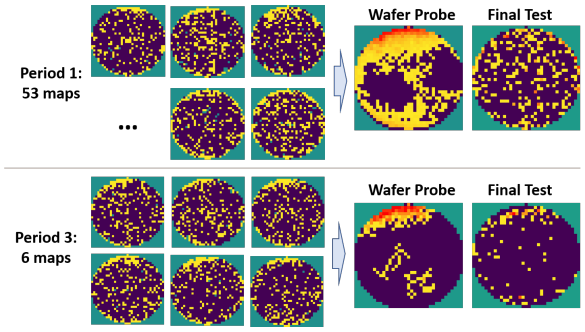


Figure 3. Further refinement by splitting the samples by month

From Figure 3, we see that the correlation is clearer on the six wafers from the Period-3 group. In addition, we found these six wafers all from the same wafer lot.

1.2. A pattern hidden in the "Edge-Local" group

In the example discussed so far, we see that the pattern class "Edge-Local" is not specific enough. The more specific pattern class "Edge-10-to-1" is useful to isolate the correlation to a few wafers. However, we knew to invoke this new pattern class "Edge-10-to-1" *after* we have seen the plot in Figure 1. The question is: Is there another pattern class that can also show a correlation and we miss it?

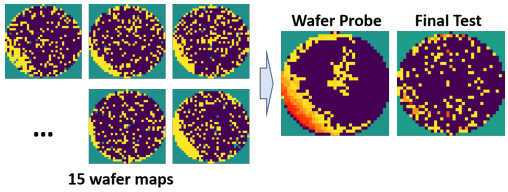


Figure 4. A hidden pattern class showing a potential correlation

The answer is yes. Figure 4 shows such an example where we might call this pattern "Edge-6-to-9". These wafers were all from the month of Period 2, and 10 of them were from the same wafer lot.

1.3. The feedback application context

Similar to the example discussed above, Figure 5 shows an example in the feedback application context. For a wafer probe's failure pattern class, we would like to see if there is an E-test parameter correlating to the failure pattern.

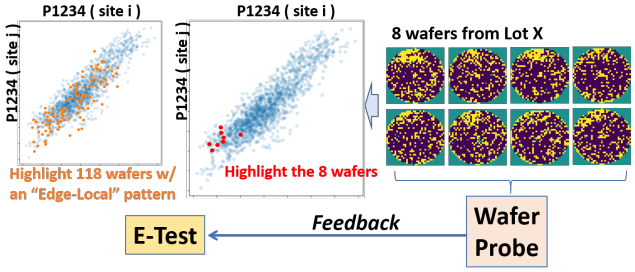


Figure 5. The feedback context considered in this work

In this example, we focus on the wafers in Period 1. The leftmost shows a plot based on 1448 wafers in the month. Among them, 118 have an "Edge-Local" pattern. These 118 wafers are highlighted as the orange dots. The plot is based on two test values of the E-test P1234 (an arbitrary name), on site i and site j. This plot shows no obvious trend for the 118 wafers (orange dots vs blue dots).

As we narrow down the wafer group selection to a new pattern "cluster spread from 10 o'clock to 12 o'clock along edge" ("ClusterEdge-10-to-12"), we found one lot (call it Lot X) containing 8 wafer maps all having the pattern. In the second E-test correlation plot, these 8 wafers are highlighted as red dots. On this second plot, we then see a clear bias of their E-test values toward the bottom-left corner. This indicates that P1234 might be associated with the "ClusterEdge-10-to-12" pattern, which is subjected to further analysis to validate (or invalidate) the finding.

1.4. The "try-and-see" analytics process

The analytics demonstrated in the feedforward and feedback contexts above can be seen as following a "try-and-see" process. The process is for searching an interesting plot serving as evidence for a finding. Each "try-and-see" analysis comprises three steps: (1) selecting a group of wafers; (2) making some plots; (3) deciding if there is an interesting finding from the plots. In each try, the extent of the interest for a plot depends on the selected group of wafers, and this extent is determined visually.

In such an analytic process, the selection of the wafer group dictates how the plots look like and hence, the extent of their interest. As a result, treating step 1 as solving a multi-class classification problem with a pre-defined set of pattern classes (as those in the prior works) is not effective. From a software tool developer perspective, step 1 requires two different kinds of capabilities: (1) When a user already knows what pattern to inspect, help the user quickly get to the desired plots. Figure 1 to Figure 3 show such an example. (2) When a user does not know what pattern to inspect, provide a good starting point. For example, getting to Figure 4 would have required such capability.

1.5. Natural language driven analytics

In this work, we envision a virtual assistant software App that supports the "try-and-see" analytics. Figure 6 depicts this view. The App takes user queries as input and produces plots as output. User queries are in natural language.

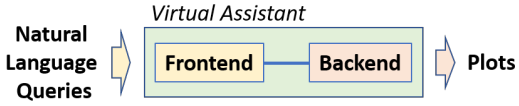


Figure 6. Virtual assistant App supporting language driven analytics

The App comprises two parts, a frontend that parses the queries into some internal executable instructions for generating a plot, and a backend that implements the functions for performing the analytics. The analytic results are stored as tables in a database that can be queried by the frontend when generating a plot. Note that implementation of the frontend is discussed in a separate work in [11]. This work focuses on the technologies in the backend analytics.

In view of the two kinds of capabilities mentioned in Section 1.4, this work discusses two analytics approaches, integrated in our backend implementation. The first is extended from our Minions approach proposed in [12]. Section 2 explains how it is used to provide a foundation for the required capabilities, and how we add the second approach called *natural language interpreter* (NLI) to enhance the analytics. Section 3 discusses the core of the wafer map interpretation problem. Then, detail for implementing the NLI is presented in Section 4. Section 5 summarizes interesting findings from test data collected on a recent product line. Section 6 concludes and points to a future work.

2. The Minions Approach

The Minions (MINiture Interactive Offset Networks) approach was developed through the works reported in [12][13]. With the approach, one neural network (NN) model is independently learned to recognize one single wafer map. Each NN recognizer is called a *Minion*. The wafer map used to train a Minion is called its *anchor*. For every pair of wafer maps, we can therefore perform mutual recognition, which results in a *recognition graph*. In this graph, every node is a wafer map. Two nodes have an edge connecting them if their recognizer recognizes each other.

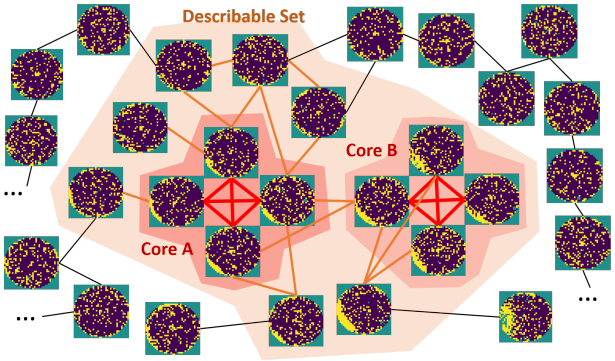


Figure 7. An example of Minions' recognition graph on wafer maps

Given a recognition graph, we can perform various analyses based on well-known graph operations. For example, starting from a node we can extract a Connected Component (CC). Figure 7 shows a CC example. It is interesting to notice that two wafer maps with a direct connection have a similar pattern. However, we cannot say that all wafer maps in the CC have a similar pattern. This is because each Minion recognizes its anchor and some variation of it. Hence, if two wafer maps are connected through multiple nodes, one can vary significantly from the other.

Given a graph, we can find its maximal cliques. In Figure 7, two maximal cliques are highlighted. In our work, we call a maximal clique a *cluster core*. The two cliques are shown as "Core A" and "Core B". Notice that within each clique, the wafer maps look very similar.

2.1. Suggesting a potential pattern class

As discussed in Section 1.4, the first step in the analytics is selecting a wafer map group. In essence, this group represents a pattern class of its own. A recognition graph like Figure 7 provides a good starting point to suggest such a pattern class. For example, we can say that a maximal clique is a potential choice of pattern class. However, as seen in Figure 7 using only the cliques would be too restricted and does not cover many wafer maps in the CC.

On the other hand, we can say that each CC is a potential choice of pattern class. However, we see that this would lead to a pattern class that is "too loose", i.e. likely two dissimilar wafer maps are included in the same class.

Given a recognition graph, using maximal clique and using CC represent two extreme cases to define pattern classes. Both are not ideal. Consequently, we need another way to choose pattern classes in between.

2.2. Primitive pattern and describable set

In our backend, a cluster core is treated as a *primitive pattern*. A pattern class can be specified and extended from a primitive pattern. Each extension is through a language interpretation based on one primitive pattern, and the interpretation result is captured in a *describable set*.

For example, given a description: "more fails spread from 6 o'clock to 9 o'clock along edge", suppose the NLI finds Core A in Figure 7 satisfying this description, and then extends from the core to find all other wafer maps also satisfying this description. The Describable Set is highlighted in the figure, which includes Core B and 7 other wafer maps. These 15 wafer maps are those shown in Figure 4 before.

2.3. Querying a describable set

In our analytics backend, a describable set corresponds to a primitive pattern and satisfies three conditions: (1) It includes the maximal clique of the primitive pattern; (2) Within itself, all wafers are connected; (3) It is describable through our *natural language interpreter* (NLI).

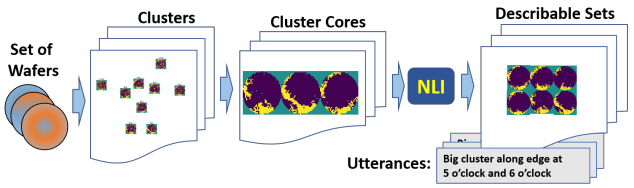


Figure 8. Attaining describable sets through NLI

Figure 8 illustrates the analytics performed in the backend. Given a set of wafer maps, first we obtain their Minions' recognition graph. From the graph, we extract connected components, and each becomes a *cluster*. Then, from each cluster we extract maximal cliques as the *cluster cores*. Based on a core, the NLI interprets a wafer map by assigning values to a set of *wafer attributes*. These wafer attributes are to be selected by an *utterance* that describes a group of wafer maps. For simplicity, in this work we rely on utterance as input query to extract the wafer group for a plotting request (detail of the frontend to use natural language queries is discussed in [11]). Given an utterance describing a pattern, the utterance is broken down into a list of constraints on the attribute values. Then, a describable set is the set of the wafers whose attribute values are based on the same core and satisfy all the constraints.

2.4. The search space

The search space comprises all describable sets which are extended from the primitive patterns. In a try, one or more describable sets can be selected based on the query. Also, a selected group can be further refined with other pattern-independent constraints, such as a yield constraint or a selected period. For example, the user can request using wafer maps only from July, with yield loss greater than 30%.

3. Explaining A Cluster Core

Figure 9 provides an example to explain more detail of the flow shown in Figure 8. Figure 9 shows a cluster, i.e. a connected component. One of the cluster cores (i.e. the maximal clique) has four wafer maps. A salient region analysis is applied to these four maps. This analysis can be based on the techniques reported in [1]. The goal is to extract an attention region on each map. Then, the four salient maps are stacked to create an *attention mask*. This mask can be obtained using a density estimation technique [14] and a threshold to include, say 80% of the density mass.

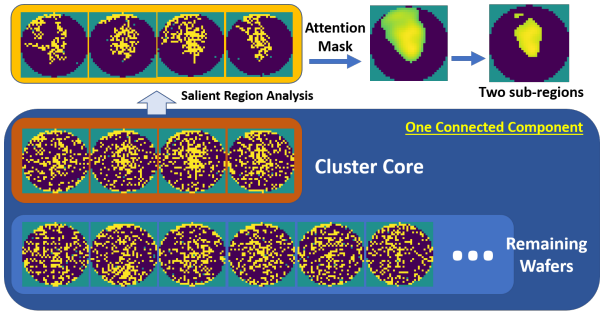


Figure 9. An example of cluster and *cluster core*

3.1. Use of the attention mask

The attention mask enables NLI to focus its interpretation on a region. This is important because it simplifies our NLI implementation. For an interpretation request, the NLI is given with a wafer map and an attention mask. For example, using the mask the NLI can simply ignore all failing dies outside. In contrast, if the NLI were given only the wafer map without a mask, the NLI has to consider all failing dies, which would be more complicated.

The attention mask has another use: It provides a way to determine how many sub-regions the NLI should separately focus on. This can be done with the gradient information included in the attention mask. For example, for the mask shown in the figure, two sub-regions are identified. NLI will then provide wafer attribute values for each sub-region.

3.2. Wafer map interpretation

For a wafer map, values of its wafer attributes depend on the attention mask in use. Since a CC can contain multiple cluster cores, each with a different attention mask, a wafer map in the CC can receive multiple interpretations from the NLI. Note that wafers in one describable set are based on one interpretation. For an input query, all satisfying describable sets are included in the group of wafer maps.

Consider the four wafer maps in the cluster core shown in Figure 9. Suppose the focus is on the top-left sub-region. For the four wafer maps, the NLI may interpret them as an "arc at x o'clock along edge" where x could be 11, 12, 10.5, and 12, respectively, depending on where the center of the arc pattern is located. These four wafer maps can all be captured in a less-constrained utterance like: "arc at 11 o'clock to 12 o'clock along edge".

For the middle sub-region, the four descriptions may all be: "big cluster at the center". Then, the capturing utterance would be the same. Furthermore, to describe the entire wafer, the NLI needs to decide a phrase that connects these two sub-regions. For example, the NLI can use a phrase like "extend to", i.e. the "arc" "extend to" the "big cluster".

4. The Natural Language Interpreter (NLI)

In the example above, we see that the NLI needs to support utterances using those *descriptive terms* such as "**arc**", "**along edge**", "**11 o'clock**", "**cluster**", "**big**", "**at the center**" and "**extend to**". Such vocabulary determines the scope of possible utterances that can be interpreted by the NLI. For building our NLI, we follow a grammatical approach [15]: (1) We use a grammar to define the scope of all possible utterances; (2) For each descriptive term, we use a software script to check for its existence on a given wafer map (i.e. setting the value of the wafer attribute).

The grammar includes the capability to support describing patterns at two levels, at individual wafer level and at multi-wafer level. Earlier we see an example where different numerical values for a descriptive term can be merged into a value range to form a less-constrained utterance. When combining two utterances that have different pattern descriptions, we can use a high-level descriptive term. For example, suppose one utterance describes an "arc at 11 o'clock along edge" and another utterance describes a "cluster at 11 o'clock along edge". In this case, we may use a high-level term "something" to capture both "arc" and "cluster". Hence, both wafer maps can be described with the utterance "something at 11 o'clock along edge".

4.1. A parsing tree example

To illustrate how the NLI follows a grammar to interpret a given wafer map, Figure 10 depicts the *parsing tree* for one wafer map (the 2nd map in the cluster core in Figure 9). For the core, there are two sub-regions. Therefore, in the parsing tree the wafer map is first partitioned into two components: $Comp_1$ and $Comp_2$. The detailed tree for the $Comp_1$ is shown in Figure 10 .

Below $Comp_1$, there are two nodes: $Comp$ and RELATION. The $Comp$ is for describing the component. The RELATION is for describing the relationship to $Comp_2$.

The $Comp$ node has two child nodes: $Check_{Arc}$ and $Descr_{Location}$. The $Check_{Arc}$ checks the type of the pattern to determine if it is an "arc". If it is, then $Descr_{Arc}$ is used to describe what kind of "arc". The $Descr_{Location}$ describes the location of the pattern. Three *attributes* are used: DIRECTION, SPREAD, REGION, and additional two are for associated PREPOSITION (PP).

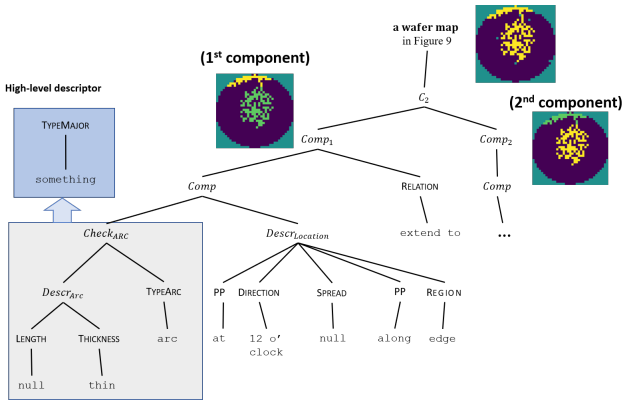


Figure 10. The parsing tree for one wafer map in Figure 9. The wafer map is described as: "**Null length thin thickness arc type at 12 o'clock direction null spread along edge extend to ...**", and at high level (using the high-level descriptor "`something`"), can be captured as: "**something at 12 o'clock direction null spread along edge extend to ...**"

The value of an attribute is determined by its corresponding software script. For example, the value of the LENGTH can be: `short` | `long` | `null`. The value of the THICKNESS can be: `thin` | `thick` | `null`. The attributes and their possible values are part of the *lexicon* in the grammar, as that exemplified in Table 1.

Sub-tree below the $Check_{Arc}$ node is highlighted with a shaded box. This is to indicate that the box can be replaced by a high-level descriptor called "`something`". With the high-level descriptor, detail of the pattern shape is ignored and the focus is on its other attributes.

In the parsing tree, the leaf nodes are values of the attributes. A *canonical utterance* can be obtained by concatenating the leaf node values and for some, their attribute names from left to right. The utterance is shown in the caption of the figure. A high-level utterance can also be constructed using the high-level descriptor "`something`".

4.2. The grammar

Our NLI uses a context-free grammar (CFG) [15] to define a formal language $\mathcal{L}_0$, which essentially models the interpretation process. A CFG is formally specified by 4-tuple (V, Σ, R, S) where V is a set of non-terminal symbols, Σ is a set of terminal symbols, $R = \{V \times (\Sigma \cup V)^*\}$ is a set of rules, and S is the starting symbol. A CFG includes a list of *production rules* that expand a non-terminal $v \in V$ into a string which can contain both non-terminals and terminals. The subset of rules that turns non-terminals into terminals, is called the *lexicon* of the grammar.

Table 1 shows our current lexicon for $\mathcal{L}_0$, which can be extended as needed. Each rule is of the form: ATTRIBUTE $\rightarrow$ {value1 | value2 | ...} (i.e. wafer attributes and values). Note that a *null* value means that there is no description for the attribute and thus can be omitted from the utterance. To use this grammar as an interpreter, a software script is implemented for each lexicon rule. The script determines which value should be selected for the wafer attribute.

Table 2 shows our current grammar rules for $\mathcal{L}_0$. The grammar rules can be used to generate various strings by recursively expanding non-terminals starting from S until every non-terminal is rewritten into a terminal according to the lexicon. All the non-terminals that trigger a rule in the lexicon (Table 1) are presented in small capital font.

TABLE 1. THE LEXICON FOR THE FORMAL LANGUAGE $\mathcal{L}_0$

TYPEARC	arc
TYPERING	ring
TYPELINE	line
TYPEDONUT	donut
TYPECLUSTER	cluster
TYPEMAJOR	something
TYPEMINOR	minor component
RELATION	and | and extend to
DIRECTION	x o'clock | left | right | upward | downward | lower left | upper left | lower right | upper right
LENGTH	short | long | *null*
THICKNESS	thin | thick | *null*
WAVINESS	straight | wavy | *null*
SIZE	small | big | huge | massive | *null*
DENSITY	solid | somewhat solid | more fails | some fails
PREPOSITION	at | near | from | to | along | around | touch | on
REGION	center | edge | in-between
SPREAD	wide | *null*
CONNECTIVITY	broken | *null*
COMPLETENESS	half | full | *null*
SIGNIFICANCE	not obvious
SUBSET	only exhibit on some wafers

The grammar rules represent the working logic in the interpretation *workflow* implemented in our NLI. For example, the first rule $(G0)$ expresses the fact that a wafer map can consist of up to two major components (C_1 for one and C_2 for two components) and some minor components (C^*). $(G5)$ to $(G7)$ are for a minor component without detailed description of its shape. $(P0)$ captures the generic procedure for describing a major component. A major component can be described in terms of its shape and location descriptors along with other optional descriptions. Specifically, each type of shape is associated with a dedicated "check" function for its determination, and every shape is associated with its unique attributes, as stated in (A), (R), (L), (D) and (C), respectively.

In addition, the internal logic of NLI ensures that patterns like an arc or line will be checked first before proceeding to a more general shape like a cluster. $(P1)$ indicates the rule for describing the location. $(P2)$ provides other options for describing the component such as its pattern significance relative to the rest of the wafer, and if it only exhibits on a subset of wafer maps in the group.

4.3. The software script

If we view the grammar as an interpretation workflow, it is not hard to see that we need two sets of software scripts to enable the interpretation. One is already mentioned above that we need a script for every lexicon rule. Another set of scripts are needed to implement other grammar rules in Table 2. For example, earlier in Section 3.1 we discuss how to use the attention mask to decide how many sub-regions to focus on. This determination can be based on finding *density peaks* in the attention mask (which can be made as a contour plot). This can be a way to implement the $(G0)$ rule in grammar, i.e. to decide whether we should follow C_1 or C_2 and whether C^* should be activated.

TABLE 2. THE GRAMMAR FOR $\mathcal{L}_0$.

	Grammar Rules	Description
(G0)	$S \rightarrow \{C_1 \vert C_2\}\, C^*$	A pattern can have one or two major components and some minor ones
(G1)	$C_1 \rightarrow Comp$	The case where there is a single major component
(G2)	$C_2 \rightarrow Comp_1\ Comp_2$	The case where there are two major components
(G3)	$Comp_1 \rightarrow Comp$ RELATION	The first major component and its relationship to the second one
(G4)	$Comp_2 \rightarrow Comp$	The second major component
(G5)	$C^* \rightarrow null$	The case where there is no other minor component
	| RELATION C	The case where there are other minor components and their relationship
(G6)	$C \rightarrow MinorComp\ C^*$	Minor components can be one or many
(G7)	$MinorComp \rightarrow$ TYPEMINOR $Descr_{Location}\ \langle opt \rangle$	A minor component is described only by its location
(P0)	$Comp \rightarrow \{Check_{Arc} \vert Check_{Ring} \vert Check_{Line} \vert Check_{Donut} \vert ... \vert Check_{Cluster}\} Descr_{Location}\ \langle opt \rangle$	The component is described by its shape and location with other options
	| TYPEMAJOR $Descr_{Location}\ \langle opt \rangle$	From high level, the component is described only by its location
(P1)	$Descr_{Location} \rightarrow$ SPREAD {PREPOSITION DIRECTION}... ...{PREPOSITION REGION}	Location is defined by the component's spread, direction and region, with additional prepositions
(P2)	$< option > \rightarrow$ SIGNIFICANCE	How obvious the component is compared to the original wafer map
	| SUBSET	The component only exhibits on a subset of wafer maps
(A0)	$Check_{Arc} \rightarrow Descr_{Arc}$ TYPEARC	
(A1)	$Descr_{Arc} \rightarrow$ LENGTH THICKNESS	An arc is defined by its length and thickness
(R0)	$Check_{Ring} \rightarrow Descr_{Ring}$ TYPELINE	
(R1)	$Descr_{Ring} \rightarrow$ THICKNESS CONNECTIVITY COMPLETENESS	A ring is defined by its thickness, connectivity, and completeness
(L0)	$Check_{Line} \rightarrow Descr_{Line}$ TYPELINE	
(L1)	$Descr_{Line} \rightarrow$ WAVINESS LENGTH THICKNESS	A line is defined by its waviness, length, and thickness
(D0)	$Check_{Donut} \rightarrow Descr_{Donut}$ TYPEDONUT	
(D1)	$Descr_{Donut} \rightarrow$ SIZE THICKNESS DENSITY COMPLETENESS	A donut is defined by its size, thickness, density, and completeness
(C0)	$Check_{Cluster} \rightarrow Descr_{Cluster}$ TYPECLUSTER	
(C1)	$Descr_{Cluster} \rightarrow$ SIZE DENSITY	A cluster can be defined by its size and density

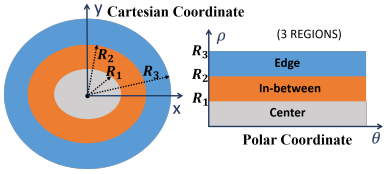


Figure 11. The definition of REGION on the wafer map

To simplify presentation, in this section we use two examples to illustrate how the scripts can be implemented. Recall in Figure 9, two sub-regions (i.e. two density peaks) are identified from the attention mask. To determine the location name of a sub-region, a wafer is divided into three areas: "edge", "in-between", and "center". An *area checker* finds which area the sub-region is in. This determination is by transforming a map into polar coordinates and dividing its radius into three areas as shown in Figure 11. The area from the wafer map center to R_1, i.e, $(\rho, \theta) = (R_1, 2\pi)$ is called the "center", the next $(R_2 - R_1, 2\pi)$ is the "in-between", and the last $(R_3 - R_2, 2\pi)$ is the "edge".

Suppose the density peak falls in the "edge" sub-region, then the component might be an "arc". To check if it is actually an "arc", we implement an *arc type checker* depicted in Algorithm 1. The input includes a threshold T_1 to determine if the component is an "arc". First, a component must have its density peak fall inside the edge region in order to be called an "arc". Then, we calculate two density estimates r, x within the component's angular spread in the edge region. The ratio $(\frac{x}{r})$ of these density estimates is compared to the threshold T_1 to determine if it is an "arc". Notice that the *arc type checker* relies on a fixed threshold.

One may raise the concern regarding the robustness of using a specific threshold, i.e. a person may see a pattern as an "arc" but the NLI fails to interpret it as an "arc". Note that our NLI does not aim to optimize with respective to such an accuracy objective. Instead, it aims to find a describable set, and as stated in Section 2.3 this set has to satisfy two conditions on the Minions' recognition graph. In this sense, the two conditions provide a check for the NLI's result.

Algorithm 1: An arc type checker

Input: Threshold: T_1, R_2, R_3
Output: Is it an Arc: True/False
Data: Wafer Map Matrix of the Component

```
edge region ← (ρ, θ) = (R3 − R2, 2π) ;      // Global
T1 = 66% ;                                  // Default
Assert the density peak is in edge region
Find the angular spread
r ← the density sum within the angular spread in
  edge region
R* ← the region (ρ, θ) = (R3, 2π) − (R2 + (R3−R2)/2, 2π)
x ← the density sum within the angular spread in R*
if x/r < T1 then
    return False ;                          // It is not an arc
return True ;                               // It is an arc
```

4.4. Canonical utterance examples

Using wafer attributes based on a given core, the NLI can interpret wafer maps with *canonical utterances*. As an example, Table 3 shows the salient wafer maps from two cores and their attention masks. The utterances interpreted

TABLE 3. SALIENT WAFER MAPS FROM A CLUSTER CORE AND THEIR CANONICAL UTTERANCES

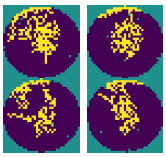
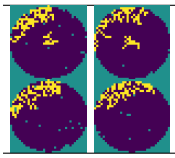
Salient Wafer Maps (Core)	Attention Mask	Canonical Utterance (1: top-left, 2: top-right, 3: bottom-left, 4: bottom-right map)
		1. null size more fails density cluster type wide spread from 9 o'clock direction to 1 o'clock direction along edge and extend to big size somewhat solid density cluster type upward direction at center. 2. null length thin thickness arc type at 12 o'clock direction null spread along edge and extend to big size somewhat solid density cluster type upper right direction at center. 3. null length thin thickness arc type at 11 o'clock direction null spread along edge and extend to huge size more fails density cluster type upper left direction at center. 4. null length thin thickness arc type at 12 o'clock direction null spread along edge and extend to big size somewhat solid density cluster type upward direction at center.
		1. null size more fails density cluster type wide spread from 9 o'clock direction to 12 o'clock direction along edge and small size cluster type upward direction at center not obvious. 2. null size some fails density cluster type wide spread from 9 o'clock direction to 1 o'clock direction on edge and small size cluster type left direction at center region not obvious. 3. null size more fails density cluster type wide spread from 9 o'clock to 1 o'clock touch edge. 4. null size more fails density cluster type wide spread from 9 o'clock direction to 2 o'clock direction on edge.

TABLE 4. EXAMPLES OF WAFER GROUPING

Cluster Core Stacked	Subgroup Stacked	Canonical Utterance
		**Something** at **11 to 12** o'clock direction along edge extend to **something around** center.
		**Something** at 11 o'clock direction wide spread along edge and **something around** center **only exhibit on some wafers**.

by the NLI are listed (1: top-left, 2: top-right, 3: bottom-left, 4: bottom-right map). Notice that wafer maps in the same cluster core can have slightly different utterances.

The high-level utterances for the two cluster cores are shown in Table 4. The table shows two stacked heatmaps for each core. The first is by stacking the four maps in the core. The second is by stacking the four maps and all the neighboring wafer maps in the describable set. A *neighboring wafer map* is directly connected to at least one wafer map in the core. This is one way we can use the recognition graph to further select a *subgroup* of wafers from a describable set. We will come back to this subgroup selection when discussing the analytic findings in Section 5.1.

The high-level descriptor "something" can be used to capture a set without detailed description of the pattern shape. For location, wafer maps in Table 3 include a variety of descriptions all related to the center region. Hence, they can be captured with another high-level descriptor "around".

For the first core, the first component can be at either 11 or 12 o'clock directions. To capture all four wafer maps, the high-level description can become "11 to 12" o'clock. Note that a "wide spread" range of two clock values can be converted into a single clock value by taking their average. In this way, the first wafer map is included in the describable set (which is a requirement).

Another consideration for a high-level utterance is that, not all wafer maps in a set have the same number of components (This is captured in ($P2$) grammar rule). The second cluster core in Table 3 exhibits this situation. In this case, an optional phrase "only exhibit on some wafers" is appended to the utterance to indicate the fact that the utterance of the second component only applies to some but not all wafers in the given set. In general, after individual canonical utterances are interpreted for wafer maps, incompatible phrases from individual wafers can be abstracted out to satisfy a given high-level utterance (e.g. "something" at the $Check_{Arc}$ and $Check_{Cluster}$ nodes to replace an "arc" phrase and a "cluster" phrase.). Note that while a high-level utterance can be used to describe a set of wafers, we make sure that a describable set always includes all wafers from the corresponding cluster core.

4.5. Paraphrasing with GPT-3

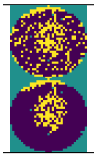
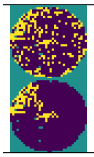
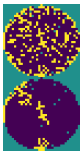
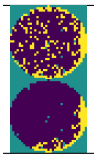
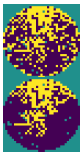
In this section, we consider converting a canonical utterance into a natural language description. This is in contrast to the work presented in [11] where the goal is to translate a natural language query into an internal representation.

To output a description more like a naturally spoken language, we can leverage the capability provided by a state-of-the-art *language model* to paraphrase canonical utterances into natural sentences. In our current implementation, the GPT-3 model [16] is used. GPT-3 is a large pre-trained neural network with 175 billion parameters. We take advantage of the *in-context learning* capability provided by GPT-3 to achieve *few-shot learning*. In other words, we teach the GPT-3 by providing a few demonstrating paraphrasing examples in a prompt window. Then, the GPT-3 will be able to convert an utterance interpreted by the NLI into a natural sentence.

In the teaching, we guided the GPT-3 by designing the input and output to follow two prefix words: "Sentence:" and "Paraphrase:", respectively. The stop word is the same as the output prefix, so that the model will stop generating once it completes one paraphrase. The GPT-3's temperature parameter controls the level of probability used to select the next word. The higher the temperature is, the more variety the chosen words will be. Since the utterances generally follow a well-defined structure according to the grammar, the temperature parameter is set to be 0.9 which allows more flexibility to paraphrase a given utterance.

To demonstrate the paraphrasing capability, Table 5 shows some examples. For each example, two maps are shown: the original wafer map and its salient region map. In each case, a canonical utterance from the NLI is shown, followed by one paraphrase given by GPT-3. Note that the GPT-3 output is obtained by giving the input "Sentence: <the utterance>" followed by the prompt "Paraphrase:". Then, the GPT-3 generates the sentence as shown.

TABLE 5. EXAMPLES OF WAFER PATTERN INTERPRETATION

Original / Salient	Canonical Utterance Interpreted by the Grammar / GPT-3 Paraphrasing Output
	**Utterance:** null length thin thickness arc type at 12 o'clock direction null spread along edge and extend to big size somewhat solid density cluster type upper right direction at center. **GPT-3:** There is a thin arc at 12 o'clock along the edge and extended to a somewhat solid big cluster located upper right at the center.
	**Utterance:** thick thickness broken connectivity half completeness ring type wide spread from 3 o'clock direction to 12 o'clock direction on edge. **GPT-3:** There shows a thick half-ring, which is not fully connected, widespread from 3 o'clock to 12 o'clock on the edge.
	**Utterance:** null size more fails density cluster type wide spread from 9 o'clock direction to 12 o'clock direction along edge and small size cluster type upward direction at center not obvious. **GPT-3:** There exists a cluster with more fails wide spread from 9 o'clock to 12 o'clock along the edge and a small cluster located upward at the center, which is not an obvious pattern.
	**Utterance:** small size cluster type at 10 o'clock direction along edge and extend to small size cluster type upper left direction near center. **GPT-3:** There exists a small cluster at 10 o'clock along the edge, which extends into a small cluster located upper left near the center.
	**Utterance:** small size cluster type at 10 o'clock direction on edge and extend to small size cluster type upper left direction near center, and minor component at 6 o'clock direction along edge. **GPT-3:** There is a small cluster at 10 o'clock along the edge which extends into a small cluster located upper left near the center, and there is also a minor component located at 6 o'clock along the edge.
	**Utterance:** long length thin thickness arc type wide spread from 2 o'clock direction to 7 o'clock direction along edge. **GPT-3:** There shows a long thin arc widespread from 2 to 7 o'clock along the edge.
	**Utterance:** null thickness broken connectivity half completeness ring type wide spread from 11 o'clock direction to 7 o'clock direction on edge and extend to null size more fails density cluster type lower right direction near center. **GPT-3:** There shows a broken half ring, widespread from 11 o'clock to 7 o'clock along the edge, which extends into a cluster with more fails located lower right near the center.
	**Utterance:** null size more fails density cluster type wide spread from 10 o'clock to 2 o'clock along the edge and extend to big size more fails density cluster type upward direction near center. **GPT-3:** There exists a cluster with more fails widespread from 10 o'clock to 2 o'clock along the edge, which extended from a big cluster with more fails upward near the center.
	**Utterance:** straight short length thick thickness line type at 8.0 o'clock direction from edge to center. **GPT-3:** There shows a short thick line which is straight, appearing at 8 o'clock pointing from the edge to the center.
	**Utterance:** big size thin thickness somewhat solid density full completeness donut type at center. **GPT-3:** There is a big yet thin donut at the center, which is somewhat solid and is fully connected

5. Application Results

Our NLI was applied to help analyze test data collected from a recent product line. The wafers were stamped with days in three months (call them Period 1,2,3). The three periods have 1448, 1383, and 74 wafers, respectively.

5.1. The feedforward application context

Figure 12 illustrates our investigation flow in the feedforward context. From the Minions' recognition graphs, we obtain a set of cluster cores. Based on each core, results from the NLI are stored in a database table. Each row in the table corresponds to a wafer map containing the wafer attribute values determined by the NLI.

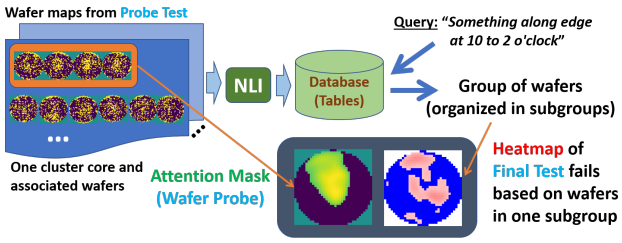


Figure 12. Flow to investigate a correlation between Wafer Probe and Final Test based on *attention mask* and final test failing *heatmap*; Each try starts with a simple query to find a group of wafers.

In a try, a query is specified to extract a group of wafers. It is important to note that, the list of primitive patterns provide a good guide for us to start a query. This was how we knew to try on the hidden pattern seen in Figure 4 before.

With a query, the describable sets are extracted as the group of wafers for plotting. In plotting, we consider arranging the wafers in *subgroups* based on primitive patterns in the group (see discussion of Table 4 before). For each subgroup, two maps are shown. The first is the *attention mask* from the primitive pattern. The second is the heatmap from final test fails based on all wafers in the subgroup. Similar to those examples shown in Figure 1 to Figure 4 before, the proximity of the patterns exhibited on these two maps is used to determine our interest to the plot.

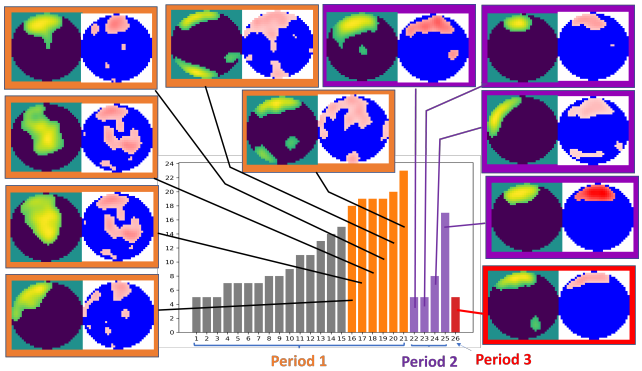


Figure 13. # of wafer maps (y-axis) in subgroups (x-axis) over three periods, which all have "something" at 10 to 2 o'clock. For each subgroup, two images are shown: (1) the left is the attention mask and (2) the right is the heatmap from final test based on all wafers in that subgroup.

Figure 13 shows a summary result arranged by subgroups along the x-axis. Each bar shows the number of

wafers in the subgroup (corresponding to a primitive pattern). Through the attention masks, we can see a pattern trend evolving over the three periods. Then, the correlation to the final test can be seen by comparing each attention mask to the corresponding final test failing heatmap.

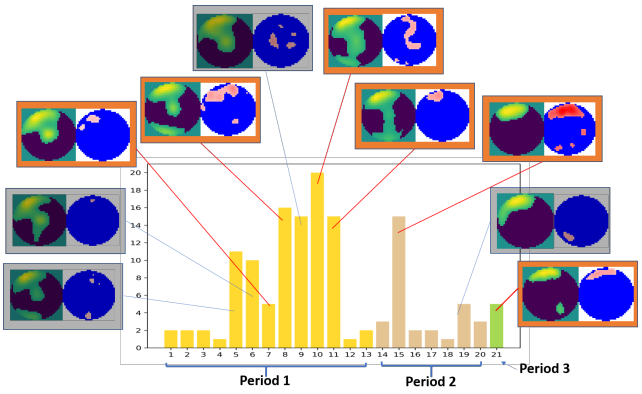


Figure 14. For the same set of wafers shown in Figure 13, this figure arrange to show day-by-day # of wafer maps over the three periods. If one day has multiple attention masks, they are stacked to show one mask.

Figure 14 shows a different summary result arranged by days along the x-axis. Each bar shows the number of wafers in that day. The plot is based on the same set of wafers used to plot Figure 13. Once we re-organize those wafers by days, both the attention mask and the heatmap for a given day need to be re-generated. As seen, in this presentation the correlation to the final test is still there for some days, but is not as apparent as that in Figure 13.

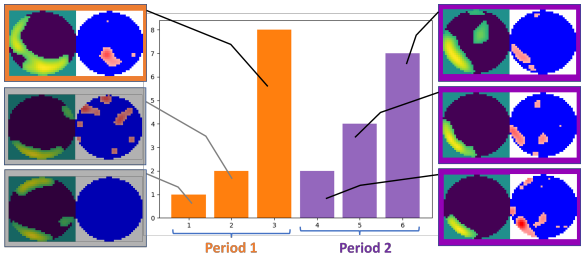


Figure 15. # of wafer maps over two periods arranged by subgroups along the x-axis, which all have "something" at 6 to 8 o'clock.

Figure 15 shows another interesting finding based on a second group and its selected subgroups of wafer maps. In this case, a yield loss threshold is used to show only those wafers with a yield loss above the threshold.

TABLE 6. SOME OTHER QUERY EXAMPLES AND RESULTS

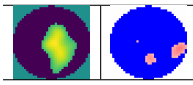
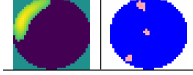
Attention	Heatmap	Query and # of wafers found
		Query: "something around center" (found 8 subgroups with a total of 51 wafer maps) (The heatmap is from a subgroup with 7 wafers)
		Query: "something widespread from 2 to 7 o'clock along edge" (found only 1 subgroup that has 7 wafer maps)
		Query: "something along edge at 8 to 10 o'clock " (found 5 subgroups with a total of 23 wafer maps) (The heatmap is from a subgroup with 4 wafers)

Table 6 shows three more examples where the correlation results are not as significant as the two examples before. Other queries like "something along edge at 4 to 6 o'clock" or "something in-between" (Center and Edge, see Figure 11) were also tried but did not find more interesting result. Note that more specific queries like "arc at 10 to 12 o'clock" or "something along edge at 11 o'clock" were also tried, but they usually found much less number of wafers, making a correlation trend over time less obvious to observe.

Overall, out of the 2905 wafers from the three periods, 2109 wafer maps were interpreted by the NLI (each belongs to some cluster). For the rest of the wafer maps, most do not have an obvious pattern and only few might be considered as having a unique pattern. If desired, they can be ranked by their yield loss to facilitate a quick manual inspection.

Example results in this section demonstrate the benefits of our analytics backend: It enables generation of various summary plots where pattern trend over time as well as correlation between two testing stages can be visualized. More importantly, the set of wafers used to generate a summary plot can be driven by a query, which in the application context provides the needed flexibility to investigate wafer map patterns from the user's own perspective.

5.2. The feedback application context

Similar to the feedforward context, in this section we use several example findings to illustrate the feedback application context. In each case, the result is reached by searching for a correlation between the described pattern class and an E-Test parameter. To refine a finding, we may use wafer lots to select a subset of wafers from the given group. In a try, we follow the same plot type as shown in Figure 5 before.

Each plot is based on two sites of an E-test parameter. Suppose there are in total N E-test parameters and k site combinations. Then, one search space contains $N \times k$ plots. To facilitate visualization, the plots are ranked according to their "bias" value, calculated as the distance between the average position of the selected wafers and the average position of the rest of the wafers. Note that in each case, we show one of the top-ranked plots.

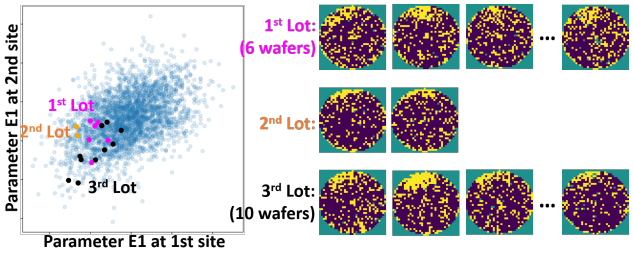


Figure 16. E-test correlation plot based on wafer maps describable by "cluster fails at direction from 11 o'clock to 12 o'clock along edge"

Figure 16 shows one finding. The 1st and 2nd sites are at the two closest locations to the pattern. Figure 17 shows another finding, based on the same E-test E1 and the same two sites in Figure 16. The descriptions between the two groups only differ by the pattern shape: one as "cluster fails" and the other as "a thin arc". Also, the 3rd lot in Figure 17 and the 3rd lot in Figure 16 are the same lot.

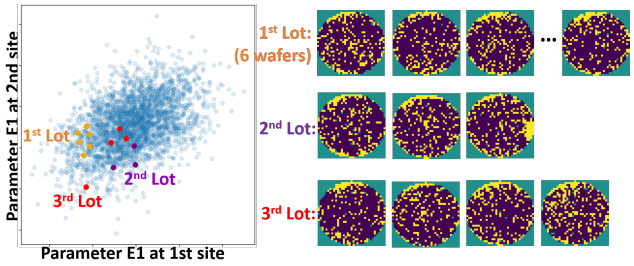


Figure 17. E-test correlation plot based on wafer maps describable by "a thin arc at direction from 11 o'clock to 12 o'clock along edge"

Figure 18 shows a finding based on a different E-test parameter E2. Again, the two sites are from the locations closest to the described pattern.

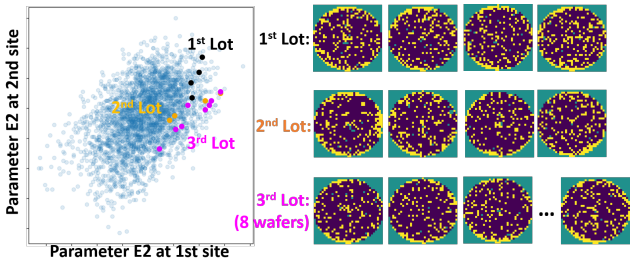


Figure 18. E-test correlation plot based on wafer maps describable by "a thin arc at direction from 5 o'clock to 7 o'clock along edge"

Figure 19 shows a finding based on E-test E3. The first site is at the center and the 2nd site is at the top. Note that this group contains two describable sets: a "big cluster" set containing wafers from the 1st and 5th lots and a "small cluster" set containing wafers from the remaining three lots.

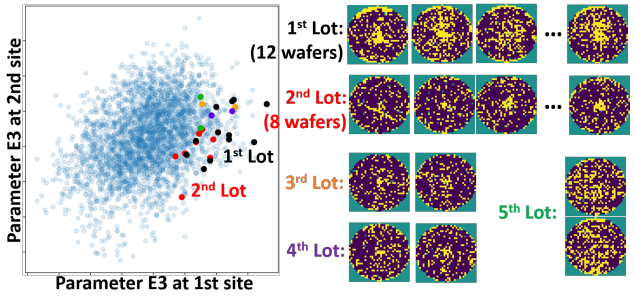


Figure 19. E-test correlation plot based on wafer maps describable by "cluster fails at or near the center"

Results shown in this section are among many interesting findings we attained for the dataset. Note that such findings provide starting points to guide further investigation. Their interpretation in view of the manufacturing process is beyond what the current analytics can provide.

6. Conclusion

In this work, we present a novel language driven approach for analyzing wafer maps. The foundation is built upon the Minions approach developed before. A natural language interpreter (NLI) is added on top of the Minions foundation. The NLI provides two major benefits: (1) It provides a flexible and yet interpretable way to group wafer maps. (2) It enables realization of a virtual assistant for analyzing wafer maps in the feedforward and feedback application contexts. The interpretation scope of the current NLI is limited by the grammar defined within. How to address the scalability issue and mitigate this limitation can be an interesting research direction in the future.

Acknowledgment This work is supported in part by National Science Foundation Grant No. 2006739 and by Semiconductor Research Corporation project No. 2020-CT-2933. The authors are thankful to Sergio Mier and Leon Wang of Qualcomm for their valuable inputs to our research.

References

[1] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer map failure pattern recognition and similarity ranking for large-scale data sets," *IEEE Tran. on Semi. Manufacturing*, vol. 28, no. 1, pp. 1–12, 2015.

[2] M. Fan, Q. Wang, and B. van der Waal, "Wafer defect patterns recognition based on optics and multi-label classification," *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016.

[3] J. Yu and X. Lu, "Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis," *IEEE Tran. on Semi. Manufacturing*, vol. 29, no. 1, pp. 33–43, 2016.

[4] a. a. Minghao Piao, "Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features," *IEEE Tran. on Semi. Manufacturing*, vol. 31, no. 2, pp. 250–257, 2018.

[5] J. Yu, "Enhanced stacked denoising autoencoder-based feature learning for recognition of wafer map defects," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 4, pp. 613–624, 2019.

[6] N. Yu, Q. Xu, and H. Wang, "Wafer defect pattern recognition and analysis based on convolutional neural network," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 4, pp. 566–573, 2019.

[7] J. Wang, Z. Yang, J. Zhang, Q. Zhang, and W.-T. K. Chien, "Adabalgan: An improved generative adversarial network with imbalanced learning for wafer defective pattern recognition," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 3, pp. 310–319, 2019.

[8] T.-H. Tsai and Y.-C. Lee, "A light-weight neural network for wafer map classification based on data augmentation," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 4, pp. 663–672, 2020.

[9] M. Saqlain, Q. Abbas, and J. Y. Lee, "A deep convolutional neural network for wafer defect identification on an imbalanced dataset in semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 3, pp. 436–444, 2020.

[10] M. B. Alawieh, D. Boning, and D. Z. Pan, "Wafer map defect patterns classification using deep selective learning," *ACM/IEEE Design Automation Conference*, 2020.

[11] Y. J. Zeng, M. J. Yang, and L.-C. Wang, "Wafer map pattern analytics driven by natural language queries," in *IEEE International Test Conferencel in Asia*, 2022.

[12] Y. J. Zeng, L.-C. Wang, and C. J. Shan, "Miniature interactive offset networks (minions) for wafer map classification," in *IEEE International Test Conferencel*. IEEE, 2021, pp. 190–199.

[13] Y. J. Zeng, L.-C. Wang, C. J. Shan, and N. Sumikawa, "Learning a wafer feature with one training sample," in *IEEE International Test Conferencel*. IEEE, 2020, pp. 1–10.

[14] N. Sumikawa, M. Nero, and L.-C. Wang, "Kernel based clustering for quality improvement and excursion detection," *IEEE International Test Conference*, 2017.

[15] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed., 2022.

[16] e. a. Tom Brown, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.