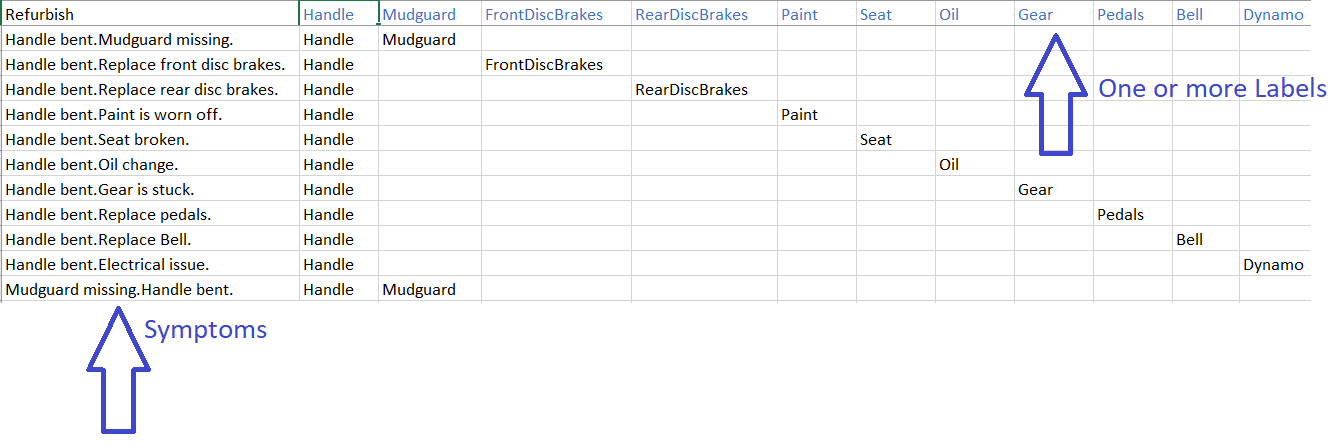
Parts Prediction – Story for far

# Traditional Approach

* Assume that a Pre-processing step breaks the WO’s Problem Description into minute symptoms.
* Multi-label classification using Google AutoML and synthetic generated data – with 11 unique Parts
* Sample data, showing the “Symptoms in a WO” & associated Labels, from a Bike Repair use-case



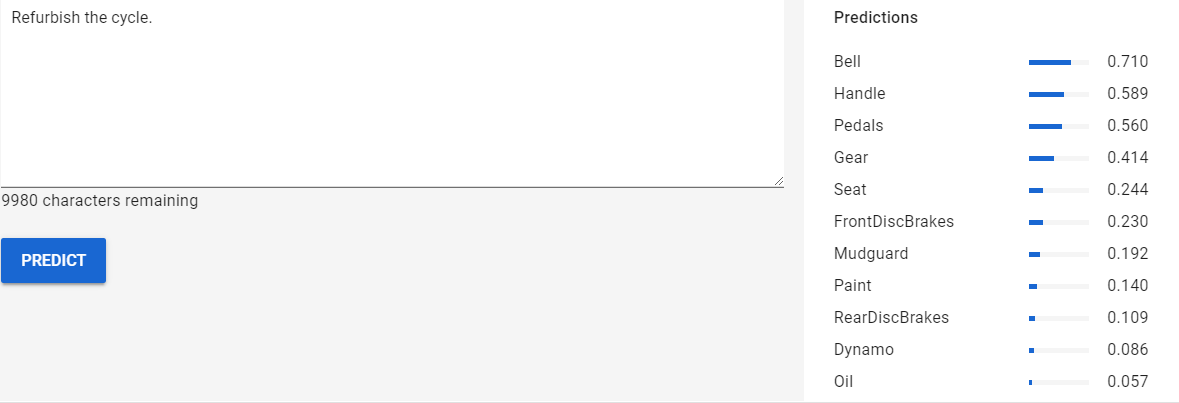
* The Training Dataset is attached.
* The Model is available at: <https://cloud.google.com/automl/ui/text/datasets/predict?dataset=TCN7214094284557293870&model=TCN4293374999740892759&project=servicemax-ai>

# Drawbacks with this approach

* Multi-Label classification is not supported by DataRobot (see attached mail). We are still trying to understand RapidMiner’s reply, but from the RapidMiner Community we gather that this feature is on the roadmap, but not currently supported.
* Approach will not scale – as the number of Parts increases. For example, the ImageNet Multi-Label dataset uses 14 million images as training data & recognizes 20,000 Labels.
* Even the above Google AutoML model cannot predict “How Many Parts” are required.
* Further, since the Google AutoML is merely throwing up some Probabilities for each Part/Label, we are unable to compare Costs for each Recommendation – like Aquant does. This might be a useful feature that enables the call center agent to choose the cheapest option.
* ML requires a lot of data for each “corner case”. Case-in-point: We have just two rows in our 4000 rows Training Dataset that relate to “Refurbishing” the Bike – All parts are to be replaced.



AutoML model does a poor job in the above scenario, recommending just 3 Parts.



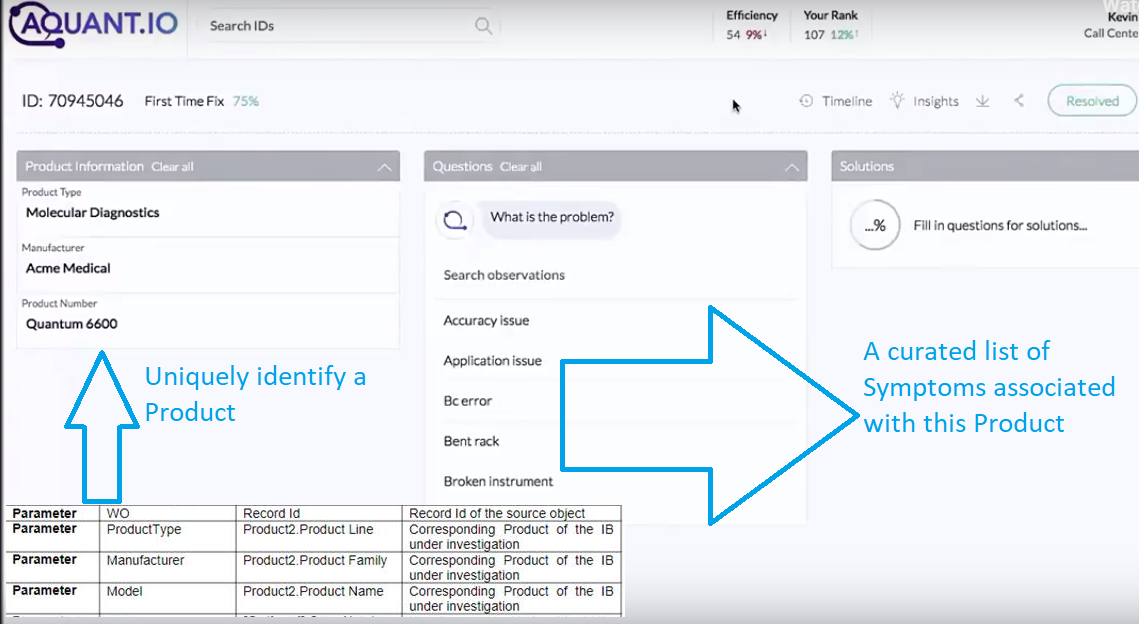
* Last but not the least, just getting data from SalesForce into a Training Data file in the above format – with each Part as a Column, can be a small project.

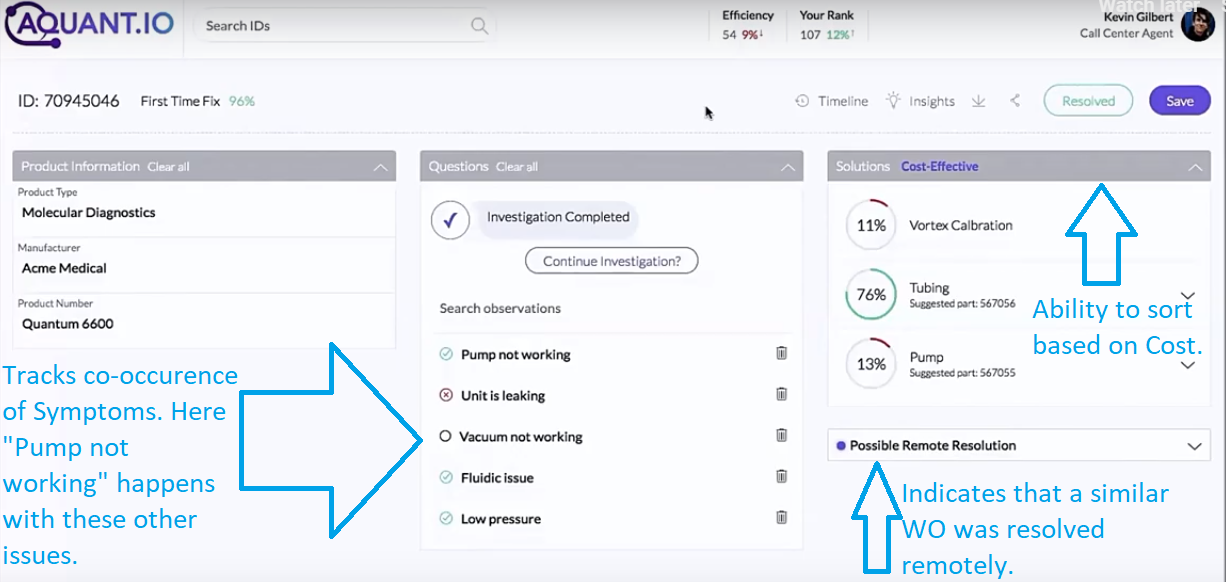
# Summary of Traditional approach

In view of the above limitations, we need to consider alternate options.

# Alternate Approach

Before getting down into the details, Aquant’s UI can guide us towards a top-down approach.





# Derive Symptoms from the Problem Description, via spaCy

A *Problem Description* like “*The Radiator is leaking, battery is overflowing, and the tubes are rusted.*” Will be broken down into 3 symptoms:

****

Further, with Semantic Similarity, duplicates are removed.

****

The result is that unstructured Text Description has now become structured.

# Master Data Schema

This Relational database enables the above Intelligent Triage. The tables are:

## ProductSymptomsAndAssociatedQuestions

Tracks the symptoms of issues in a Product

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Manufacturer | ProductType | Model | SymptomId | SymptomText | Question | Reviewed (ONLY Reviewed records will show in the UI) |
| Philips | MRI | V1 | 1 | Radiator leaking | Is the radiator leaking? | N |
| Philips | MRI | V1 | 2 | Battery overflowing | Is the Battery overflowing? | N |
| Philips | MRI | V1 | 3 | Tubes rusted | Are the Tubes rusted? | N |

## WOSymptomCoOccurence

Tracks co-occurrence across WOs and Symptoms. This table helps in:

1. Finding co-occurring symptoms from WOs, given a SymptomId

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manufacturer | ProductType | Model | WO-ID | SymptomId |
| Philips | MRI | V1 | WO-1 | 1 |
| Philips | MRI | V1 | WO-1 | 1 |
| Philips | MRI | V1 | WO-1 | 2 |
| Philips | MRI | V1 | WO-1 | 2 |
| Philips | MRI | V1 | WO-2 | 1 |
| Philips | MRI | V1 | WO-2 | 1 |
| Philips | MRI | V1 | WO-2 | 3 |
| Philips | MRI | V1 | WO-2 | 3 |

## PartsSymptomCoOccurence

1. Given a Symptom, suggest Parts with Probability %.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Manufacturer | ProductType | Model | WO-ID | PartId | NumParts | RemotelyTriaged |
| Philips | MRI | V1 | WO-1 | Part1 | 2 | Y |
| Philips | MRI | V1 | WO-1 | Part2 | 1 | Y |
| Philips | MRI | V1 | WO-1 | Part1 | 2 | Y |
| Philips | MRI | V1 | WO-1 | Part2 | 1 | Y |
| Philips | MRI | V1 | WO-2 | Part1 | 1 | N |
| Philips | MRI | V1 | WO-2 | Part2 | 2 | N |
| Philips | MRI | V1 | WO-2 | Part1 | 1 | N |
| Philips | MRI | V1 | WO-2 | Part2 | 2 | N |

## ToDo

1. Given a Symptom and PartId, suggest average number of Parts needed.
2. Given a Final set of Symptoms, suggest if it can be Remotely resolved (based on WO History)

## PartsMaster

Tracks Parts consumed by a WO. Included only for comparison with Aquant. Duplication of Parts information between Salesforce schema & AI schema can be avoided, as it can lead to issues if the two tables are out of sync.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Manufacturer | ProductType | Model | PartId | PartName | UnitCost |
| Philips | MRI | V1 | 1 | Tube1 | 2 |
| Philips | MRI | V1 | 2 | Tube2 | 2 |
| Philips | MRI | V1 | 3 | Tube3 | 1 |
| Philips | MRI | V1 | 4 | Tube4 | 1 |

# Semantic Search Google BigQuery Table (a separate micro-service)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Manufacturer | ProductType | Model | WO-ID | SymptonText | SymptomTextVectorForm | Cost | ResolvedViaRemoteTriage | Status |
| Philips | MRI | V1 | WO-1 | Radiator leaking. Battery overflowing. Tubes rusted | VECTOR1 | 100 | N | CLOSED |
| Philips | MRI | V1 | WO-2 | Tubes rusted. | VECTOR2 | 200 | N | CLOSED |
| Philips | MRI | V1 | WO-3 | Radiator leaking. | VECTOR3 | 300 | N | CLOSED |
| Philips | MRI | V1 | WO-4 | Battery overflowing. | VECTOR4 | 0 | Y | CLOSED |

# Data-flows

## Fetch Symptoms for a machine (List required by UI)

*Select from ProductSymptomsAndAssociatedQuestions where Manufacturer =? and ProductType=? And Model =?*

## Given SymptomText, fetch Top N most similar WOs (BigQuery)

Semantic Search in BigQuery returns WOIds along with Top-Level info (required for sorting based on Cost & knowing if a similar WO was resolved Remotely)

*Select WO-ID, Cost, ResolvedViaRemoteTriage from BigDataTable where Manufacturer =? and ProductType=? and Model =? and CosineSimilarTo(SymptomText)*

## Given Top N most similar WOs and an input SymptonTextId, find the OTHER issues that can co-occur

Select SymptomId, count(\*) from WOSymptomPartCoOccurence where WOId IN (WO-1, WO-2, WO-3) and SymptomId NOT IN (Sympton1, Symptom2) //AS THESE ARE ALREADY CHOSEN

Now, which-ever Symptom occurred most – Ask the corresponding Question (in the UI), by querying *ProductSymptomsAndAssociatedQuestions.* If User chooses *Yes,* append SymptomText & re-query BigQuery.

Continue until Symptoms are exhausted.

## On Work-Order close

Update BigQueryTable with Top level info like Cost, ResolvedViaRemoteTriage and Status

## Setup Initial List of most commonly occurring Symptoms (for a given Product)

Run every *Product Description* through the Preprocessing Service to get the Symptom list. This Symptom list will be refined by removing Duplicates. Here again, Text Embeddings are very useful – as they can help in automating removing of Duplicates. Corresponding Questions can be generated or created manually.