VOLVO

Ideas for vehicle signal set matching

Peter Winzell, Jon Seneger
Volvo Cars, Sunnyvale

Modern connected vehicle networks handle several thousand different data signals. Data signals such as wheel pulses, battery level, location and window positions. Vehicle and vehicle data signals, and their corresponding identification, has historically been proprietary and not exposed to 3rd parties, with new use cases stemming from smart cities, autonomous vehicles, electrical vehicles, and various other connected services, there is a need for a standardized way of describing vehicles and vehicle data. The W3C [1] and COVESA [2] have together been trying to resolve this by proposing a web standard for vehicle signal data VISS [3] [4], VSS [5]. The standard will facilitate data access and promote innovation in the vehicle data space. Mapping and matching signals to this standard is a tedious and time-consuming task. We are suggesting ideas for a candidate ranking algorithm based on the lexical elements of signals names, their semantic description, type, and unit. We also use a knowledge graph - Neo4j [6] - to represent candidate matches.

Problem definition

To start defining the signal set matching problem we start with a definition where we view the solution as bipartite graph [7] where the vertices represent signals from two separate signal sets and the edges a match between two signals[1].

$let\ U = \{A, B\}, where\ A = \{S_1, \dots, S_n\}, B = \{S'_1, \dots, S'_k\}$,

$represent\ signal\ set\ A\ and\ B.$

$Find\ a\ mapping\ M: A \cap B \cup A - B,$

$M: G = (A, B, E)\ is\ a\ bipartite\ graph$

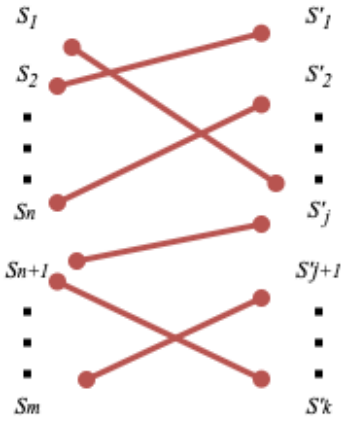

Figure 1 Signal set graph

To find this mapping we divide the problem in two:

1) For $A \cap B$ a ranking algorithm is suggested where a candidate matching

   $m = \{S_j \in A, S'_j \in B\}, is\ assigned\ a\ score\ p, 0 < p < 1.$

2) For A – B adding missing signals could be done when

   $p \rightarrow 0.$

[1] Throughout this article our aim was to explore mappings for VSS

In common for vehicles signal sets are that they are typically associated with attributes such as name, description, unit, range, and datatype. Given that a match exists we use this fact to build a ranking algorithm.

Algorithm:

$$p = \frac{\sum_{i=1}^{n} f_i w_i}{n}, where\ f_i \in [0,1] and\ w_i \in [0,1],$$

n *is the number of attribute matching functions.*

We used 4 attribute functions $f_1, \dots, f_4$

With the weights $w_1, \dots, w_4$ we tune the algorithm.

Normalization

Vehicle signals are typically exposed on the CAN [8], Flexray [9], and ethernet buses/interfaces. To facilitate the solution, we first transform and clean the vehicle data sets into a human readable JSON format. For CAN based signal sets we are able to use cantools [10], an open source based set of tools for CAN based networks. For Flexray we developed a customized parser. VSS is open sourced and has a set of tools that can be used for this purpose.

```
[
 {
  "signaltype": 0,
  "signame": "Vehicle.ADAS.ABS.IsActive",
  "lextokens": [
   "Vehicle",
   "ADAS",
   "ABS",
   "Is",
   "Active"
  ],
  "description": "Indicates if ABS is enabled. True =
Enabled. False = Disabled.",
  "datatype": "boolean",
  "unit": ""
 },
…

]
```

Figure 2 Data normalization file format

Candidate matching algorithm

We defined 4 different attribute functions that use the name, description, unit, and datatype as the input. By adding weights for each of these functions we can tune the individual mapping functions against each other, e.g., defining a behavior where name matching is more important than datatype matching.

Name matching – f_1

Signal names often consist of different lexical elements that describes its functionality and origin in the vehicle. By applying the Levenshtein [10] distance on a lexical element vector we can measure the word distance between two signal names and use that for our name matching algorithm.

Flexray	VehSpd	Veh	Spd
VSS	Vehicle.Speed	Vehicle	Speed

$v_A = \{"Veh","Spd"\}$
$v_B = \{"Vehicle","Speed"\}$

We define each lexical element as a token, which means that VehSpd it tokenized as {“Veh”,”Spd”} and Vehicle.Speed as{“Vehicle”,”Speed”}.We are also able to generate a token similarity graph and use that to expand/replace tokens from one signal set. If the Levenhstein distance between a lexical element of set A and another from set B is sufficiently small, we can use that for expansion.

$$L(l_A, l_B) \rightarrow \varepsilon, where\ \varepsilon\ is\ suffiently\ small$$

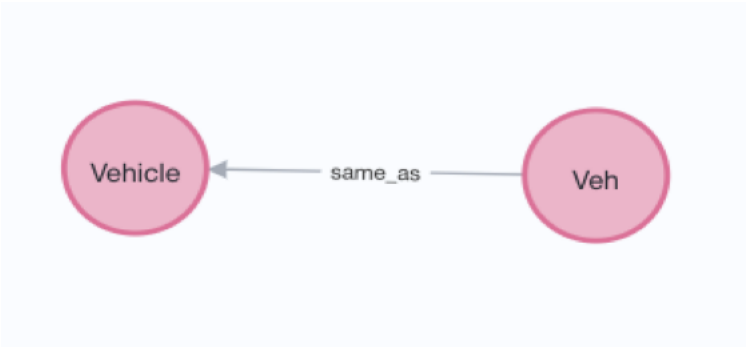

Figure 3 Token similarity graph

The expansion of one signal set using matching tokens makes the actual name matching more accurate. Given the example above the AUTOSAR signal VehSpd =[“Veh”,”Spd”] would expand into VehSpd=[“Vehicle”,”Speed”] which then would result in an exact match in VSS.

for a = A' **do**
 for b = B’ **do**
 if $if\ lev(a,b) < \varepsilon$
 $A[a] = b$

Figure 4 token expansion

The order in which a signal name is divided into different tokens have impact on using Levehnstein word distance. The algorithm needs to avoid the order bias as much as possible, since the actual naming convention is unknown from one signal set to another.

Set A	Name	Tokenized
S1	VehSpd	“Veh”,”Spd”}
S2	SpdVeh	{“Spd”,”Veh”}
Set B	Name	Tokenized
S3	Vehicle.Speed	{“Vehicle”,”Speed”}
$L(S1,S3) = 6$	L is the Levenshtein function	
$L(S2,S3) = 12$		

Figure 5 token order problem

So, by selecting the minimum distant token and adding that to a sum of token distances we are avoiding order bias. A measured token cannot be measured again and thus both tokens are removed from the token sets when selected. For quantification we also keep track of the maximum length of selected tokens, and we get a value between [0,1] by dividing our sum of minimum token distances with 2 * the maximum token lengths.

Input: string a and string b. Where a and b are signal names from disjunctive signal sets A, B.
$A_e$is the expansion look up table generated for all tokens in A.

Output:
$value\ p\ [0..1], where\ p = 1 - \frac{\sum \text{mind}}{\sum maxd}.$
$mind\ is\ the\ sum\ of\ minimum\ levenshtein\ distance\ divided\ by\ the\ possible\ max$

for A **do**
A' = tokenize(A)
for B **do**
B' = tokenize(B)

// token expansion replacement assignment
For $\boldsymbol{t_a = A'}$ **do**
if $A_e[t_a]$ exist
$A'[t_a] = A_e[t_a]$

$min_d = MAX\_INT$
$min_s = 0$
$max_s = 0$

for $A' \& B' \neq 0$ **do**
for $\forall\ t_a \in A'$ **do**
for $\forall t_b \in B'$ **do**
$\boldsymbol{C = lev(t_a, t_b)}$
If $C < mind$
$mind = C$
$maxd = Max_{lev}(t_a,\ t_b)$
$Ca = Copy(t_a)$
$Cb = Copy(t_b)$

min_s = min_s + min_d
max_s = max_s + max_d

from A' **delete** Ca
from B' **delete** Cb

If s_length(A') > 0
for $\forall t_a$
$L(t) = stringlength(t_a)$
$min_s = min_s + L(t)$
$max_s = max_s + 2 \cdot L(t)$

ret $1 - \frac{min_s}{max_s}$

Figure 6 minimum token Levenshtein algorithm

Signal description distance - f_2

The next attribute function takes the description text from each signal and applies a neural network model trained by using Word2vec [11] [12]. Word2vec uses a model where we vectorize words in a multidimensional space and can measure the distance between words where words that have a similar semantic meaning, such as vehicle and car, are placed close to one another in this space.

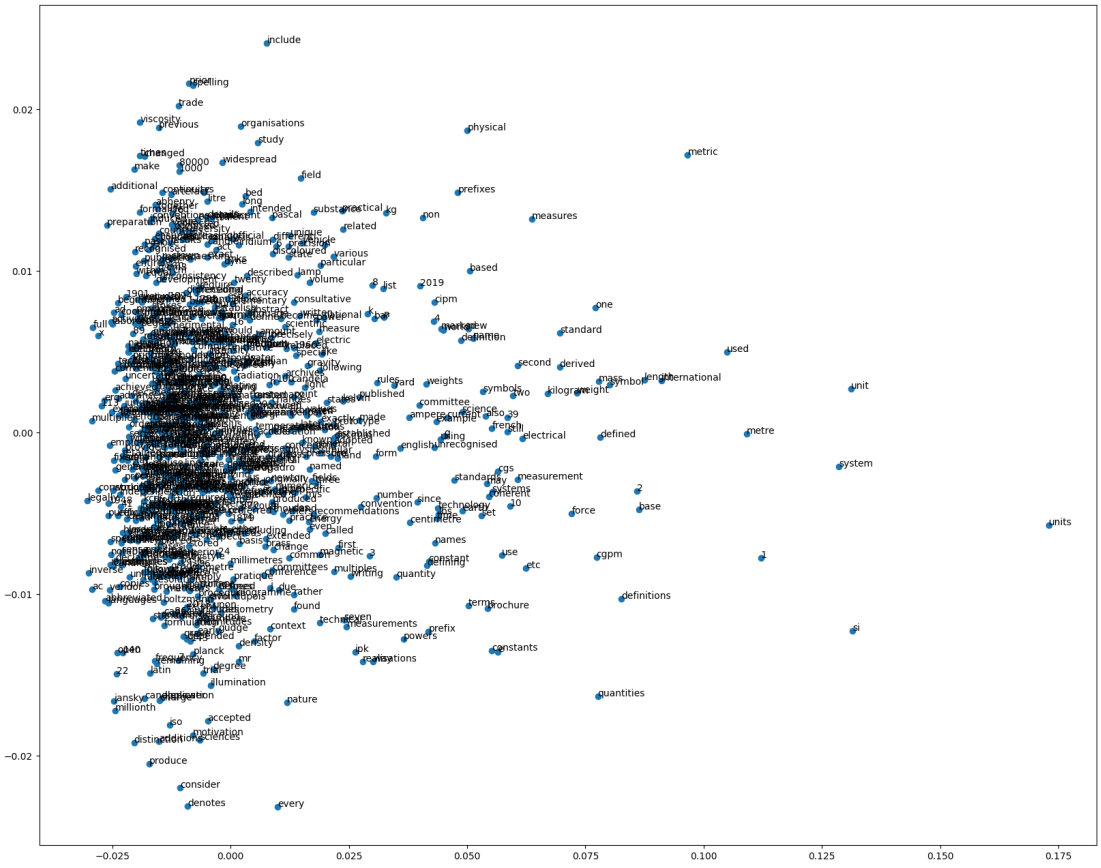

Figure 7 vectorized words

The model was tailored for automotive, e.g., going thru texts where the content is within the automotive space. We use cosine similarity [13] - figure 8 - to predict the word distance between two texts, in our case the description of the signals from two disjunctive signal sets.

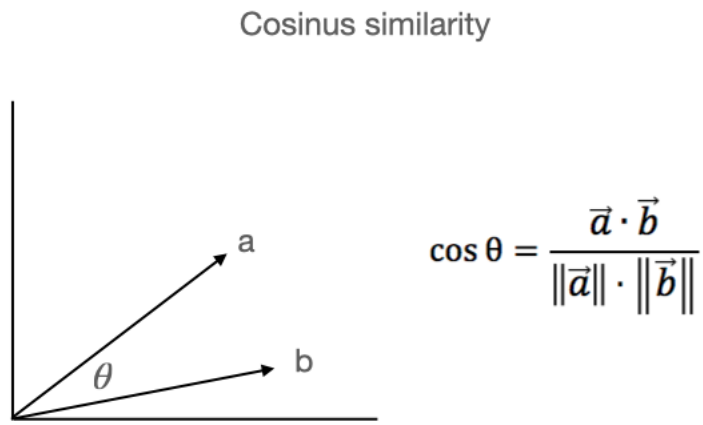

Figure 8 cosine similarity

The modification to the actual output value p is that we assume that all descD >= K should return the similarity score 0.

```
Signal a.desc = “returns the speed of the vehicle in meter per second”
Signal b.desc = “returns the velocity of the car in miles per hour”

descD = word2vec.cosineD(a.desc, b.desc)
```

Figure 9 description comparision

Unit similarity - f_3

We know that m/s and km/h are related the idea is to build a unit similarity graph. This will let us set a number to how two units relate to each other.

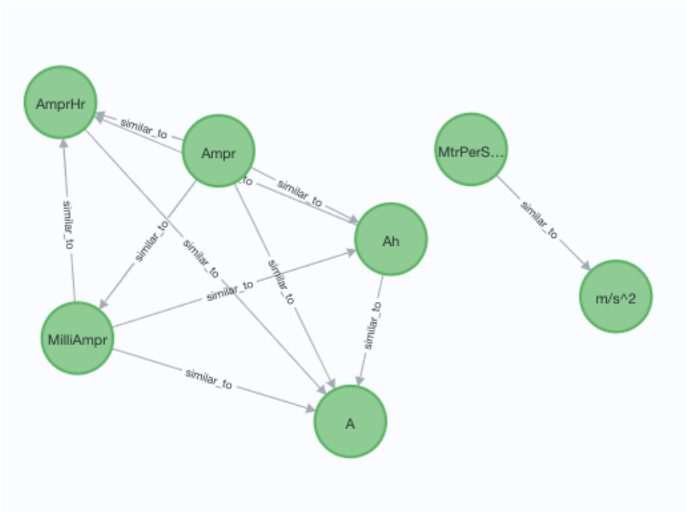

Figure 10 Unit similarity graph

Figure 10 shows a subset of the graph where the *similar_to* edge is associated with a number between 0..1.

Data type similarity - f_4

Signal sets are delivered with different data types. The signals also consist of range in which we should expect a signal value to reside in. For example if we are talking about angels they should have a value from 0..360 deg, or 0..2π. This type of information is also used when calculating the overall similarity between two signals.

We are using a similar knowledge graph as the unit similarity to return a value between 0..1.

Ranking algorithm summary

4 different signal similarity functions $f_1, \ldots, f_4$ which are using the attributes *name, description, unit and data type* is used to get a score p [0..1], where a score of 1 would be a perfect match. We also associate a weight $w_1, \ldots, w_4$ with each function since we need to value if name matching is more worth than say unit similarity.

Results

For validation a random selection of matches should give a good estimate in how well the algorithm performs. The validation of these samples must be subjected to a human validator. Below is a table of non-random selected samples although promising an actual statistical analysis would not show that the algorithm in its initial state is able to generally match the signal sets. (Original names have been obfuscated for proprietary reasons)

Results - MATCH p=()-[r:match]->() WHERE r.value > .5 RETURN p

Signalname	Description	Unit	Datatype	P
WinPosFrntRe***	"Set request for Front Right window position"	Perc	A_UNIT8	
Vehicle.Cabin.Door.Row1.Right.Window.Position	"Window position 0 = fully closed 100 - fully opened"	percent	uint8	
				0.79
EngSpdMa***	"Maximum engine speed value for engine speed meter.."	Rpm	A_UINT16	
Vehicle.Powertrain.CombustionEngine.Engine.Speed	"Engine speed measured as rotations per minute."	rpm	uint16	
				0.79
DstFromClstrForOb***	"Indicated odometer value in resolution 0.1km …"	KiloMtr	A_UINT32	
Vehicle.Powertrain.Transmission.TravelledDistance	"Odometer reading, total distance travelled during the lifetime of the transmission."	km	float	
				0.6
PosnLatInPosnFro***	"GNSS positioning data provided by …"	Deg	A_INT32	
Vehicle.CurrentLocation.Latitude	"Current latitude of vehicle."	degrees	double	
				0.61
WinPosnReLeSt***	"Status of rear left Windows activated via voice…"	Perc		
Vehicle.Cabin.Door.Row2.Left.Shade.Position	"Position of side window blind. 0 = Fully retracted. 100 = Fully deployed"	percent		
				0.79
VehSpdLgt***	"Vehicle speed longitudinal based on wheel speed sensors and longitudinal acceleration…"	MtrPerSec	A_UINT16	
Vehicle.Speed	"Vehicle speed"	km/h	float	
				0.55
EscCtrlIn***	"Indicates that an Electronic stability control …"	NoUnit	A_UINT8	
Vehicle.VehicleIndentification.Model	"Vehicle model"		String	
				0.29

Figure 11 Signal matching results

We could also use observations of real data from each set to further enhance these ideas – a behavioral approach to signal mapping. By tuning and tailoring the algorithm, and for example giving more room for description comparison using Word2vec, we should be able to improve and move beyond the first state.

Bibliography

[1] "W3C," [Online]. Available: https://www.w3.org.

[2] "COVESA," Connected Vehicle Systems Alliance, [Online]. Available: https://www.covesa.global.

[3] "VISS Versions 2 - Core," Vehicle Information Service Specification, [Online]. Available: https://www.w3.org/TR/viss2-core/.

[4] «VISS Version 2 - Transport,» W3C, 2022. [En ligne]. Available: https://w3c.github.io/automotive/spec/VISSv2_Transport.html.

[5] "Vehicle Signal Specification," COVESA, 2022. [Online]. Available: https://covesa.github.io/vehicle_signal_specification/.

[6] neo4j, neo4j, 2022. [Online]. Available: https://neo4j.com.

[7] "MathWorld," MathWorld, [Online]. Available: https://mathworld.wolfram.com/BipartiteGraph.html.

[8] "CiA," CiA.org, [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/.

[9] "ISO," Flexray Consortium, 2009. [Online]. Available: https://www.iso.org/standard/59804.html.

[10] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," Soviets Physics Doklady, Vol. 10 p 707, 1966.

[11] K. C. G. C. J. D. Tomas Mikolov, "Efficient Estimation of Word Representations in Vector Space," Cornell Univ, 2013.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," 2013.

[13] "ScienceDirect," ScienceDirect, [Online]. Available: https://www.sciencedirect.com/topics/computer-science/cosine-similarity.