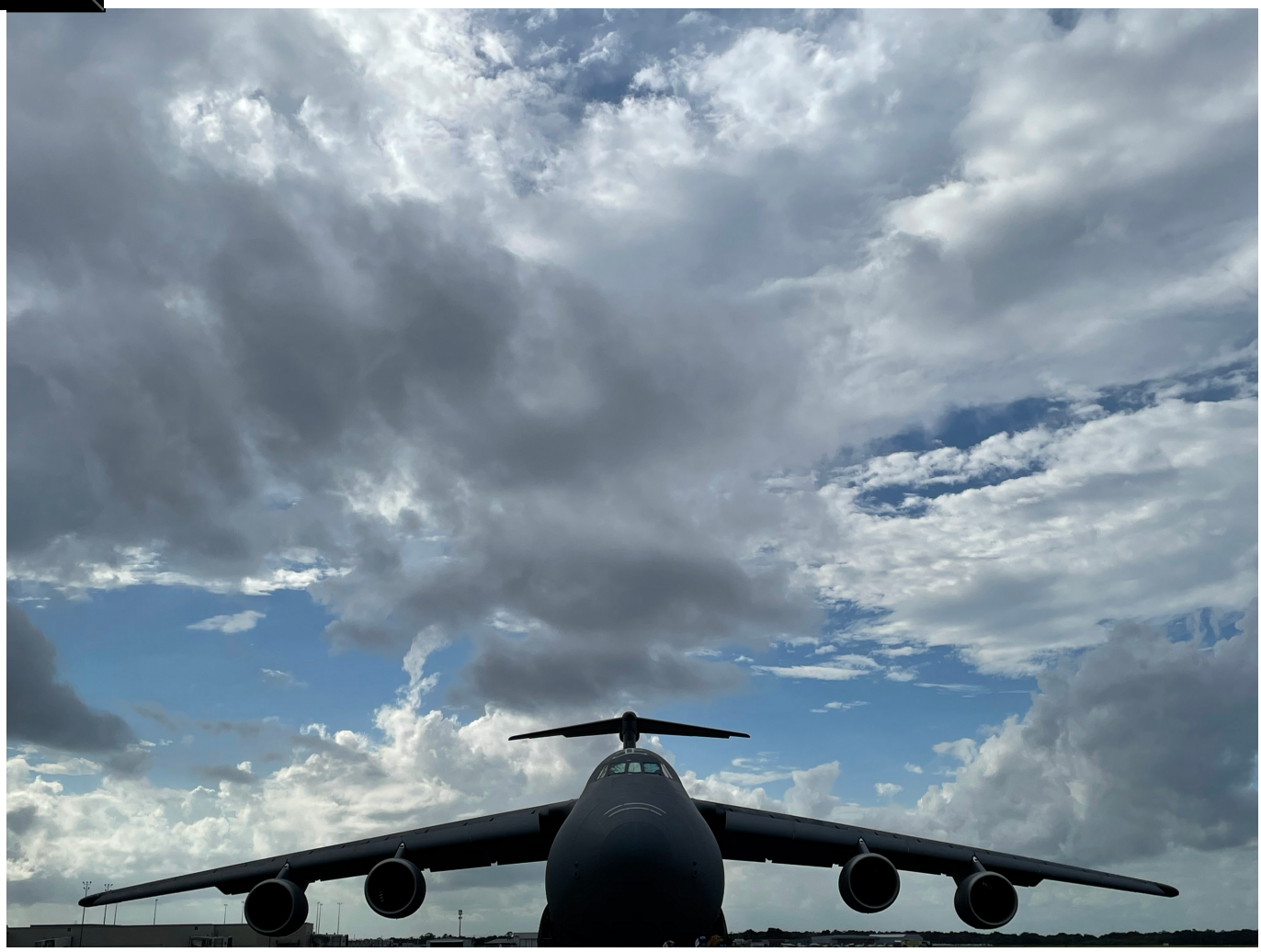
Hashed Categoric Encodings with Automunge

High cardinality, low overhead



“Hashing is a form of cryptography in which a message is transformed into an encoded

representation.”

```
=> '0f44cb01d838c981156d9f0c030159fb'
```

In common practice hashing may be used to validate voracity of a message’s sender, such as e.g. by comparing a received hash of a bank account number to a hash of that number on file without having to transmit the actual account number through a channel which may be exposed to an eavesdropper. Thus, a hashing is a deterministic transform where consistently received data will return a consistent encoding. There are exceptions though, as in cases where a randomness seed may be incorporated into a hash by a “salting” in order to further mask transmissions from eavesdroppers.

In the context of a machine learning workflow, a hashing algorithm may become useful for purposes of encoding a high cardinality categoric feature set because they enable consistent translations from strings to integers without the use of a conversion dictionary to serve as basis, which for sets with a large number of unique entries may become unwieldy.

```
['unique', 'entries', 'in', 'feature', 'set']

=> {'entries' : 0, 'feature' : 1, 'in' : 2, 'set' : 3, 'unique' : 4}
```

The use of a hashing algorithm to perform these type of conversions is known as “the hashing trick” [1, 2]. The hashing trick works by first translating entries to a hashing, followed by conversion to a numeric representation.

```
"Hashing is a form of cryptography in which a message is transformed
into an encoded representation."

=> 202956135984492237198036400469003043 79
```

Consistent with the hashing serving as origin, this numeric form will be deterministic such that a consistent message will return a consistent numeric representation. Now in

order to convert this numeric representation into an integer encoding, we can simply divide by a configurable integer and let the remainder serve as the encoding, where this configurable integer represents the range of embedding space, e.g. if we divide by 10 then we will have capacity for 10 distinct encodings.

```
"Hashing is a form of cryptography in which a message is transformed
into an encoded representation."

=> 9
```

Because the hashing trick relies on this division operation, there is possibility that different unique entries may result in an encoding overlap, with such probability getting larger with smaller embedding space. Since the intended use is for high cardinality sets, this is deemed an acceptable tradeoff.

```
['unique', 'entries', 'in', 'feature', 'set']

=> [6, 2, 8, 7, 7]
```

Automunge offers a few variants on hashing transforms. In the base configuration 'hash' transformation category, unique entry strings are given some preprocessing in order to separately encode distinct words found within entries.

```
“Unique entries (in feature set).”

=> [6, 2, 8, 7, 7]
```

This is conducted by stripping special characters and extracting words based on space separators, where special characters, space separator, and embedding size are all configurable parameters. The extracted encodings are returned in separate columns, with the number of columns derived based on the largest entry in the train set, and entries with fewer words are padded out with zeros. The encoding space size may either be configured or based on a heuristic of twice the vocabulary size. The hashing may be

by the native python hash function as default or a parameter may activate a md5 hash basis. A salt parameter may also be designated to perturb encoding basis for privacy preservation. For an upstream uppercase conversion to ignore case configurations the 'Uhsh' transformation category can also be applied.

	pre-transform feature set	processed via 'Uhsh' (integer encoding of extracted words by hashing trick with uppercase conversion)				
	stringset	stringset_UPCS_hash_0	stringset_UPCS_hash_1	stringset_UPCS_hash_2	stringset_UPCS_hash_3	stringset_UPCS_hash_4
0	'Unique entries (in feature set).'	3	9	9	1	15
1	'More unique entries'	4	3	9	0	0
2	'Another string in feature set.'	7	5	9	1	15

For encodings based on the full entries without extraction of words and special characters, a variation is available as 'hsh2' (or 'Uhs2' with uppercase conversion). This version is loosely comparable to ordinal encoding, where the tradeoffs include potential for encoding overlaps, and lack of inversion support as there is no conversion dictionary assembled. A similar transform is available to return binary encodings instead of integer encodings with 'hs10' / 'Uh10'.

	pre-transform feature set	'hsh2'	'hs10'			
	string	string_hash	string_hs10_0	string_hs10_1	string_hs10_2	string_hs10_3
0	'unique'	5	1	0	0	1
1	'entries'	1	0	1	0	1
2	'in'	3	0	0	0	0
3	'feature'	5	1	0	0	0
4	'set'	8	0	1	0	1
5	'feature'	5	1	0	0	0
6	'set'	8	0	1	0	1

The whole point of Automunge is that data transformations can be fit to properties of the train set for consistent efficient processing of subsequent data. While the hashing transforms are unique for our categoric library in that a conversion dictionary is not required for the hashing, in our implementation there are still train set properties that are saved in a populated dictionary and applied to subsequent test data, like train set embedding space size (either user specified or inferred under heuristic) and also the various parameters when activated. These hashing transformation categories are

[redacted]

intended for use in high cardinality categoric sets, as an alternative to those transforms previously discussed in our paper [redacted] which are better suited for categoric feature sets with a bounded range of unique entries.

Automunge: We make machine learning easy.

Acknowledgments

A thank you owed to the authors of [redacted] for their introduction to the hashing trick. The Automunge implementation was partly inspired by review of the Tensorflow keras_preprocessing hashing_trick function.

References

[1] John Moody. Fast Learning in Multi-Resolution Hierarchies. NIPS Proceedings, 1989 (Link)

[2] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, Josh Attenberg. Feature Hashing for Large Scale Multitask Learning. ICML Proceedings, 2009 (Link)

[redacted]

[redacted]

[redacted]