StatisticsAndDataMining

Statistics and Data Mining in Hive

This page is the secondary documentation for the slightly more advanced statistical and data mining functions that are being integrated into Hive, and especially the functions that warrant more than one-line descriptions.

ngrams() and context_ngrams(): N-gram frequency estimation

N-grams are subsequences of length **N** drawn from a longer sequence. The purpose of the `ngrams()` UDAF is to find the `k` most frequent n-grams from one or more sequences. It can be used in conjunction with the `sentences()` UDF to analyze unstructured natural language text, or the `collect()` function to analyze more general string data.

Contextual n-grams are similar to n-grams, but allow you to specify a 'context' string around which n-grams are to be estimated. For example, you can specify that you're only interested in finding the most common two-word phrases in text that follow the context "I love". You could achieve the same result by manually stripping sentences of non-contextual content and then passing them to `ngrams()`, but `context_ngrams()` makes it much easier.

Use Cases

1. (ngrams) Find important topics in text in conjunction with a stopword list.
2. (ngrams) Find trending topics in text.
3. (context_ngrams) Extract marketing intelligence around certain words (e.g., "Twitter is ___").
4. (ngrams) Find frequently accessed URL sequences.
5. (context_ngrams) Find frequently accessed URL sequences that start or end at a particular URL.
6. (context_ngrams) Pre-compute common search lookaheads.

Usage

```
SELECT context_ngrams(sentences(lower(tweet)), 2, 100 [, 1000]) FROM twitter;
```

The command above will return the top-100 bigrams (2-grams) from a hypothetical table called `twitter`. The `tweet` column is assumed to contain a string with arbitrary, possibly meaningless, text. The `lower()` UDF first converts the text to lowercase for standardization, and then `sentences()` splits up the text into arrays of words. The optional fourth argument is the **precision factor** that control the tradeoff between memory usage and accuracy in frequency estimation. Higher values will be more accurate, but could potentially crash the JVM with an OutOfMemory error. If omitted, sensible defaults are used.

```
SELECT context_ngrams(sentences(lower(tweet)), array("i","love",null), 100, [, 1000]) FROM twitter;
```

The command above will return a list of the top 100 words that follow the phrase "i love" in a hypothetical database of Twitter tweets. Each `null` specifies the position of an n-gram component to estimate; therefore, every query must contain at least one `null` in the context array.

Note that the following two queries are identical, but `ngrams()` will be slightly faster in practice.

```
SELECT ngrams(sentences(lower(tweet)), 2, 100 [, 1000]) FROM twitter;
SELECT context_ngrams(sentences(lower(tweet)), array(null,null), 100, [, 1000]) FROM twitter;
```

Example

```
SELECT explode(ngrams(sentences(lower(val)), 2, 10)) AS x FROM kafka;
{"ngram":[of","the],"estfrequency":23.0}
{"ngram":[on","the],"estfrequency":20.0}
{"ngram":[in","the],"estfrequency":18.0}
{"ngram":[he","was],"estfrequency":17.0}
{"ngram":[at","the],"estfrequency":17.0}
{"ngram":[that","he],"estfrequency":16.0}
{"ngram":[to","the],"estfrequency":16.0}
{"ngram":[out","of],"estfrequency":16.0}
{"ngram":[he","had],"estfrequency":16.0}
{"ngram":[it","was],"estfrequency":15.0}
```

```
SELECT explode(context_ngrams(sentences(lower(val)), array("he", null), 10)) AS x FROM kafka;
{"ngram":[was],"estfrequency":17.0}
{"ngram":[had],"estfrequency":16.0}
{"ngram":[thought],"estfrequency":13.0}
{"ngram":[could],"estfrequency":9.0}
{"ngram":[would],"estfrequency":7.0}
{"ngram":[lay],"estfrequency":5.0}
{"ngram":[s],"estfrequency":4.0}
{"ngram":[wanted],"estfrequency":4.0}
{"ngram":[did],"estfrequency":4.0}
{"ngram":[felt],"estfrequency":4.0}
```

histogram_numeric(): Estimating frequency distributions

Histograms represent frequency distributions from empirical data. The kind that is referred to here are histograms with variable-sized bins. Specifically, this UDAF will return a list of (x,y) pairs that represent histogram bin centers and heights. It's up to you to then plot them in Excel / Gnuplot / Matlab / Mathematica to get a nice graphical display.

Use Cases

1. Estimating the frequency distribution of a column, possibly grouped by other attributes.
   2. Choosing discretization points in a continuous valued column.

   ### Usage

```
SELECT histogram_numeric(age) FROM users GROUP BY gender;
```

The command above is self-explanatory. Converting the output into a graphical display is a bit more involved. The following Gnuplot command should do it, assuming that you've parsed the output from `histogram()` into a text file of (x,y) pairs called `data.txt`.

```
plot 'data.txt' u 1:2 w impulses lw 5
```

Example

```
SELECT explode(histogram_numeric(val, 10)) AS x FROM normal;
{"x":-3.6505464999999995,"y":20.0}
{"x":-2.7514727901960785,"y":510.0}
{"x":-1.7956678951954481,"y":8263.0}
{"x":-0.9878507685761995,"y":19167.0}
{"x":-0.2625338380837097,"y":31737.0}
{"x":0.5057392319427763,"y":31502.0}
{"x":1.2774146480311135,"y":14526.0}
{"x":2.083955560712489,"y":3986.0}
{"x":2.9209550254545484,"y":275.0}
{"x":3.674835214285715,"y":14.0}
```