

Text Processing with Hive

Chapter 12.1



Text Processing with Hive

- **How to use Hive's most important string functions**
- **How to format numeric values**
- **How to use regular expressions in Hive**
- **What n-grams are and why they are useful**
- **How to estimate how often words or phrases occur in text**

Chapter Topics

Text Processing with Hive

- **Overview of Text Processing**
- Important String Functions
- Using Regular Expressions in Hive
- Sentiment Analysis and n-grams

Text Processing Overview

- **Traditional data processing relies on highly-structured data**
 - Carefully curated information in rows and columns
- **What types of data are we producing today?**
 - Free-form notes in electronic medical records
 - Application and server log files
 - Social network connections
 - Electronic messages
 - Product ratings
- **These types of data also contain great value**
 - But extracting it requires a different approach

Chapter Topics

Text Processing with Hive

- Overview of Text Processing
- **Important String Functions**
- Using Regular Expressions in Hive
- Sentiment Analysis and n-grams

Basic String Functions

- Hive supports many string functions often found in RDBMSs

Function Description	Example Invocation	Input	Output
Convert to uppercase	UPPER (name)	Bob	BOB
Convert to lowercase	LOWER (name)	Bob	bob
Remove whitespace at start/end	TRIM (name)	└─ Bob ─┐	Bob
Remove only whitespace at start	LTRIM (name)	└─ Bob ─┐	Bob└─
Remove only whitespace at end	RTRIM (name)	└─ Bob ─┐	└─ Bob
Extract portion of string *	SUBSTRING (name, 0, 3)	Samuel	Sam

* Unlike SQL, starting position is zero-based in Hive

Parsing URLs with Hive

- **Hive offers built-in support for parsing Web addresses (URLs)**
- **The following examples assume the following URL as input**
`-http://www.example.com/click.php?A=42&Z=105`

Example Invocation	Output
<code>PARSE_URL(url, 'PROTOCOL')</code>	<code>http</code>
<code>PARSE_URL(url, 'HOST')</code>	<code>www.example.com</code>
<code>PARSE_URL(url, 'PATH')</code>	<code>/click.php</code>
<code>PARSE_URL(url, 'QUERY')</code>	<code>A=42&Z=105</code>
<code>PARSE_URL(url, 'QUERY', 'A')</code>	<code>42</code>
<code>PARSE_URL(url, 'QUERY', 'Z')</code>	<code>105</code>

Numeric Format Functions

- **Hive offers two functions for formatting a number**
 - Simple: `FORMAT_NUMBER` (0.10.0 and later)
 - Versatile: `PRINTF` (0.9.0 and later)

Example Invocation	Input	Output
<code>FORMAT_NUMBER(commission, 2)</code>	2345.519728	2,345.52
<code>PRINTF("\$%1.2f", total_price)</code>	356.9752	\$356.98
<code>PRINTF("%s owes \$%1.2f", name, amt)</code>	Bob, 3.9	Bob owes \$3.90
<code>PRINTF("%.2f%%", taxrate * 100)</code>	0.47314	47.31%

- **Caution: avoid storing precise values as floating point numbers!**
 - These functions are best used for formatting results

Splitting and Combining Strings

- **CONCAT** combines one or more strings
 - The `CONCAT_WS` variation joins them with a separator
- **SPLIT** does nearly the opposite
 - Difference: return value is `ARRAY<STRING>`

Example Invocation	Output
<code>CONCAT('alice', '@example.com')</code>	<code>alice@example.com</code>
<code>CONCAT_WS(' ', 'Bob', 'Smith')</code>	<code>Bob Smith</code>
<code>CONCAT_WS('/', 'Amy', 'Sam', 'Ted')</code>	<code>Amy/Sam/Ted</code>
<code>SPLIT('Amy/Sam/Ted', '/')</code>	<code>["Amy", "Sam", "Ted"] *</code>

** Representation of **ARRAY<STRING>***

Converting Array to Records with EXPLODE

- The **EXPLODE** function creates a record for each element in an array
 - An example of a *table generating function*
 - The alias is required when invoking table generating functions

```
hive> SELECT people FROM example;  
Amy, Sam, Ted
```

```
hive> SELECT SPLIT(people, ',') FROM example;  
["Amy", "Sam", "Ted"]
```

```
hive> SELECT EXPLODE(SPLIT(people, ',')) AS x FROM example;  
Amy  
Sam  
Ted
```

Chapter Topics

Text Processing with Hive

- Overview of Text Processing
- Important String Functions
- **Using Regular Expressions in Hive**
- Sentiment Analysis and n-grams

Regular Expressions

- A regular expression (*regex*) matches a pattern in text
 - Useful when exact matching is not practical

Regular Expression	String (matched portion in bold)
Dualcore	I wish Dualcore had 2 stores in 90210.
\\d	I wish Dualcore had 2 stores in 90210.
\\d{5}	I wish Dualcore had 2 stores in 90210 .
\\d\\s\\w+	I wish Dualcore had 2 stores in 90210.
\\w{5,9}	I wish Dualcore had 2 stores in 90210.
.?\\.	I wish Dualcore had 2 stores in 9021 0 .
.*\\.	I wish Dualcore had 2 stores in 90210 .

Hive's Regular Expression Functions

- **Hive has two important functions that use regular expressions**
 - REGEXP_EXTRACT returns the matched text
 - REGEXP_REPLACE substitutes another value for the matched text
- **These examples assume that `txt` has the following value**
 - It's on Oak St. or Maple St in 90210

```
hive> SELECT REGEXP_EXTRACT(txt, '(\d{5})', 1)
      FROM message;
```

```
90210
```

```
hive> SELECT REGEXP_REPLACE(txt, 'St.?\s+', 'Street ')
      FROM message;
```

```
It's on Oak Street or Maple Street in 90210
```

Regex SerDe

- **We sometimes need to analyze data that lacks consistent delimiters**
 - Log files are a common example of this

```
05/23/2013 19:45:19 312-555-7834 CALL_RECEIVED ""
05/23/2013 19:45:23 312-555-7834 OPTION_SELECTED "Shipping"
05/23/2013 19:46:23 312-555-7834 ON_HOLD ""
05/23/2013 19:47:51 312-555-7834 AGENT_ANSWER "Agent ID N7501"
05/23/2013 19:48:37 312-555-7834 COMPLAINT "Item not received"
05/23/2013 19:48:41 312-555-7834 CALL_END "Duration: 3:22"
```

- **RegexSerDe will read records based on supplied regular expression**
 - Allows us to create a table from this log file

Creating a Table with Regex SerDe (1)

Log excerpt

```
05/23/2013 19:45:19 312-555-7834 CALL_RECEIVED ""  
05/23/2013 19:48:37 312-555-7834 COMPLAINT "Item not received"
```

RegexSerDe

```
CREATE TABLE calls (  
    event_date STRING,  
    event_time STRING,  
    phone_num STRING,  
    event_type STRING,  
    details STRING)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'  
WITH SERDEPROPERTIES ("input.regex" =  
    "([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) \"([^\"]*)\"");
```

- Each pair of parentheses denotes a field
 - Field value is text matched by pattern within parentheses

Creating a Table with Regex SerDe (2)

Log excerpt

```
05/23/2013 19:45:19 312-555-7834 CALL_RECEIVED ""
05/23/2013 19:48:37 312-555-7834 COMPLAINT "Item not received"
```

RegexSerDe

```
CREATE TABLE calls (
  event_date STRING,
  event_time STRING,
  phone_num STRING,
  event_type STRING,
  details STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES ("input.regex" =
  "([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) \"([^\"]*)\"");
```

Table excerpt

event_date	event_time	phone_num	event_type	details
05/23/2013	19:45:19	312-555-7834	CALL_RECEIVED	
05/23/2013	19:45:37	312-555-7834	COMPLAINT	Item not received

Regex SerDe in Older Versions of Hive

- **The Regex SerDe wasn't formally part of Hive prior to 0.10.0**
 - It shipped with Hive, but was part of the “hive-contrib” library
- **To use Regex SerDe in 0.9.x and earlier versions of Hive**
 - Add this JAR file to Hive
 - Change the SerDe's package name, as shown below

```
CREATE TABLE calls (
```

The package name used in older versions is slightly different:

`org.apache.hadoop.hive.contrib.serde2.RegexSerDe`

```
details STRING,
```

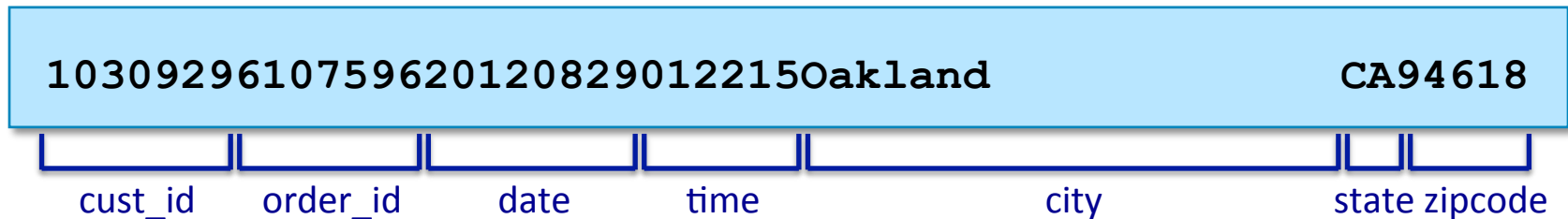
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
```

```
WITH SERDEPROPERTIES ("input.regex" =
```

```
"([^\ ]*) ([^\ ]*) ([^\ ]*) ([^\ ]*) \"([^\"]*)\"");
```

Fixed-Width Formats in Hive

- Many older applications produce data in fixed-width formats



- Unfortunately, Hive doesn't directly support these
 - But you can overcome this limitation by using `RegexSerDe`
- Caveat: all fields in `RegexSerDe` are of type **STRING**
 - May need to cast numeric values in your queries

Fixed-Width Format Example

1030929610759620120829012215Oakland

CA94618

Input data

RegexSerDe

```
CREATE TABLE fixed (  
  cust_id STRING,  
  order_id STRING,  
  order_dt STRING,  
  order_tm STRING,  
  city STRING,  
  state STRING,  
  zip STRING)  
ROW FORMAT SERDE  
  'org.apache.hadoop.hive.serde2.RegexSerDe'  
WITH SERDEPROPERTIES ("input.regex" =  
  "(\\d{7})(\\d{7})(\\d{8})(\\d{6})(.{20})(\\w{2})(\\d{5})");
```

cust_id	order_id	order_dt	order_tm	city	state	zipcode
1030929	6107596	20120829	012215	Oakland	CA	94618

Chapter Topics

Text Processing with Hive

- Overview of Text Processing
- Important String Functions
- Using Regular Expressions in Hive
- **Sentiment Analysis and n-grams**

Parsing Sentences into Words

- Hive's **SENTENCES** function parses supplied text into words
- Input is a string containing one or more sentences
- Output is a two-dimensional array of strings
 - Outer array contains one element per sentence
 - Inner array contains one element per word in that sentence

```
hive> SELECT txt FROM phrases WHERE id=12345;
```

```
I bought this computer and really love it! It's very fast and  
does not crash.
```

```
hive> SELECT SENTENCES(txt) FROM phrases WHERE id=12345;
```

```
[["I", "bought", "this", "computer", "and", "really", "love", "it"],  
 ["It's", "very", "fast", "and", "does", "not", "crash"]]
```

Sentiment Analysis

- **Sentiment analysis is an application of text analytics**
 - Classification and measurement of opinions
 - Frequently used for social media analysis
- **Context is essential for human languages**
 - Which word combinations appear together?
 - How frequently do these combinations appear?
- **Hive offers functions that help answer these questions**

n-grams

- **An n-gram is a word combination (n=number of words)**
 - Bigram is a sequence of two words (n=2)
- **n-gram frequency analysis is an important step in many applications**
 - Suggesting spelling corrections in search results
 - Finding the most important topics in a body of text
 - Identifying trending topics in social media messages

Calculating n-grams in Hive (1)

- **Hive offers the NGRAMS function for calculating n-grams**
- **The function requires three input parameters**
 - Array of strings (sentences), each containing an array (words)
 - Number of words in each n-gram
 - Desired number of results (top-N, based on frequency)
- **Output is an array of STRUCT with two attributes**
 - `ngram`: the n-gram itself (an array of words)
 - `estfrequency`: estimated frequency at which this n-gram appears

Calculating n-grams in Hive (2)

- The NGRAMS function is often used with the SENTENCES function
 - We also used LOWER to normalize case
 - And EXPLODE to convert the resulting array to a series of rows

```
hive> SELECT txt FROM phrases WHERE id=56789;  
This tablet is great. The size is great. The screen is  
great. The audio is great. I love this tablet! I love  
everything about this tablet!!!
```

```
hive> SELECT EXPLODE(NGRAMS(SENTENCES(LOWER(txt)), 2, 5))  
      AS bigrams FROM phrases WHERE id=56789;  
{ "ngram": ["is", "great"], "estfrequency": 4.0 }  
{ "ngram": ["great", "the"], "estfrequency": 3.0 }  
{ "ngram": ["this", "tablet"], "estfrequency": 3.0 }  
{ "ngram": ["i", "love"], "estfrequency": 2.0 }  
{ "ngram": ["tablet", "i"], "estfrequency": 1.0 }
```

Finding Specific n-grams in Text

- **CONTEXT_NGRAMS** is similar, but considers only specific combinations
 - Additional input parameter: array of words used for filtering
 - Any **NULL** values in the array are treated as placeholders

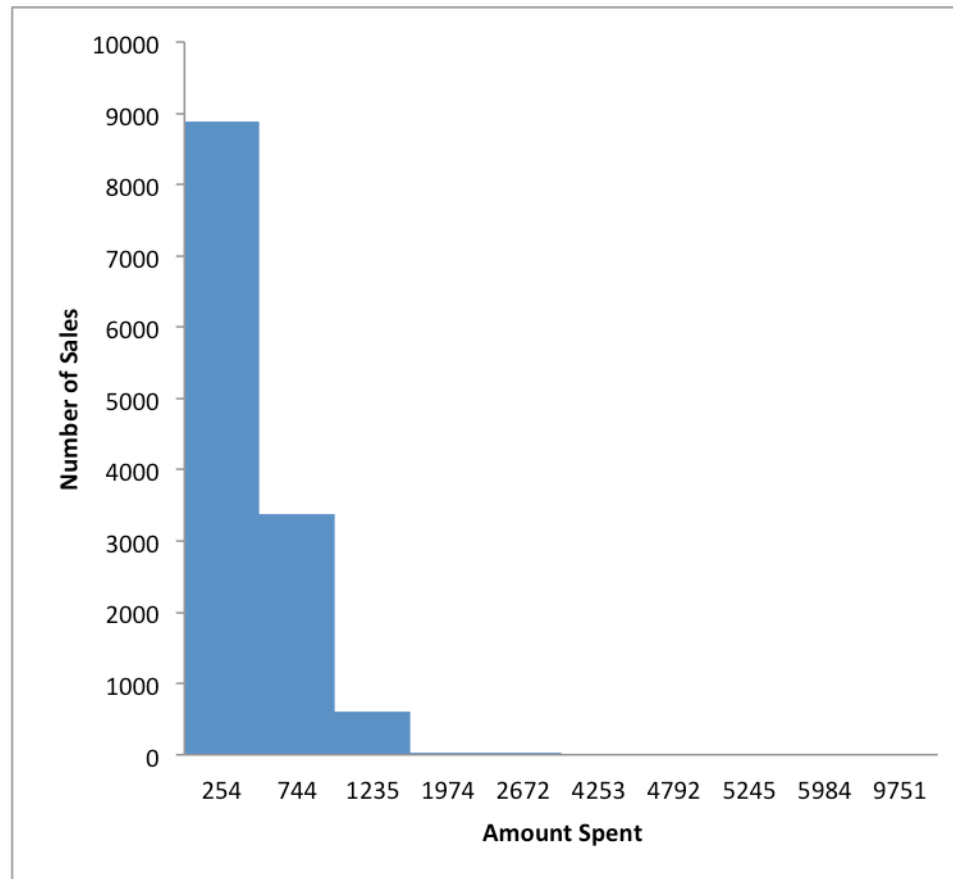
```
hive> SELECT txt FROM phrases
      WHERE txt LIKE '%new computer%';
```

```
My new computer is fast! I wish I'd upgraded sooner.
This new computer is expensive, but I need it now.
I can't believe her new computer failed already.
```

```
hive> SELECT EXPLODE(CONTEXT_NGRAMS(SENTENCES(LOWER(txt)),
      ARRAY("new", "computer", NULL, NULL), 4, 3)) AS ngrams
      FROM phrases;
{"ngram":["is","expensive"],"estfrequency":1.0}
{"ngram":["failed","already"],"estfrequency":1.0}
{"ngram":["is","fast"],"estfrequency":1.0}
```

Histograms

- **Histograms illustrate how values in the data are distributed**
 - This helps us estimate the overall shape of the data distribution



Calculating Data for Histograms

- **HISTOGRAM_NUMERIC** creates data needed for histograms
 - Input: column name and number of “bins” in the histogram
 - Output: coordinates representing bin centers and heights

```
hive> SELECT EXPLODE(HISTOGRAM_NUMERIC(  
    total_price, 10)) AS dist FROM cart_orders;  
{ "x": 25417.336745023003, "y": 8891.0 }  
{ "x": 74401.5041469194, "y": 3376.0 }  
{ "x": 123550.04418985262, "y": 611.0 }  
{ "x": 197421.12500000006, "y": 24.0 }  
{ "x": 267267.53846153844, "y": 26.0 }  
{ "x": 425324.0, "y": 4.0 }  
{ "x": 479226.38461538474, "y": 13.0 }  
{ "x": 524548.0, "y": 6.0 }  
{ "x": 598463.5, "y": 2.0 }  
{ "x": 975149.0, "y": 2.0 }
```

Import this data into
charting software to
produce a histogram

Essential Points

- **Most data produced these days lacks rigid structure**
 - Text processing can help us analyze loosely-structured data
- **The `SPLIT` function creates an array from a string**
 - `EXPLODE` creates individual records from an array
- **Hive has extensive support for regular expressions**
 - You can extract or substitute values based on patterns
 - You can even create a table based on regular expressions
- **An n-gram is a sequence of words**
 - Use `NGRAMS` and `CONTEXT_NGRAMS` to find their frequency