Introduction of Feature Hashing

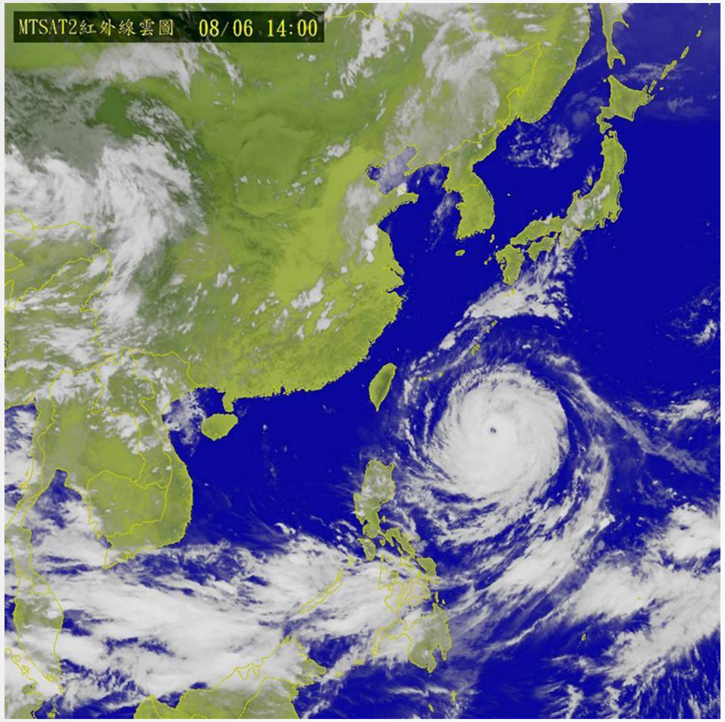
Create a Model Matrix via Feature Hashing with a Formula Interface

Wush Wu
Taiwan R User Group

Wush Chi-Hsuan Wu

- Ph.D. Student
  - Display Advertising
  - Large Scale Machine Learning
- Familiar tools: R, C++, Apache Spark

Taiwan, a Mountainous Island



source: http://www.nbcnews.com

- Highest Mountain: Yushan (3952m)

Taipei City and Taipei 101

source: https://c2.staticflickr.com/6/5208/5231215951_c0e0036b17_b.jpg

New Landmark of Taipei

Taiwan R User Group

http://www.meetup.com/Taiwan-R

An Example of FeatureHashing

Sentiment Analysis

- Provided by Lewis Crouch
- Show the pros of using `FeatureHashing`

Dataset: IMDB

- Predict the rating of the movie according to the text in review.
- Training Dataset: 25000 reviews
- Binary response: positive or negative.

```
## [1] 1
```

- Cleaned review

```
## [1] "kurosawa is a proved humanitarian this movie is totally about people living in"
## [2] "poverty you will see nothing but angry in this movie it makes you feel bad but"
## [3] "still worth all those who s too comfortable with materialization should spend 2"
## [4] "5 hours with this movie"
```

Word Segmentation

```
##  [1] ""                 "kurosawa"         "is"
##  [4] "a"                "proved"           "humanitarian"
##  [7] ""                 "this"             "movie"
## [10] "is"               "totally"          "about"
## [13] "people"           "living"           "in"
## [16] "poverty"          ""                 "you"
## [19] "will"             "see"              "nothing"
## [22] "but"              "angry"            "in"
## [25] "this"             "movie"            ""
## [28] "it"               "makes"            "you"
## [31] "feel"             "bad"              "but"
## [34] "still"            "worth"            ""
## [37] "all"              "those"            "who"
## [40] "s"                "too"              "comfortable"
## [43] "with"             "materialization" "should"
## [46] "spend"            "2"                "5"
## [49] "hours"            "with"             "this"
## [52] "movie"            ""
```

Word Segmentation and Feature Extraction

ID	MESSAGE	HATE	BAD	LIKE	GRATUITOUS
"5814_8"	TRUE	TRUE	TRUE	TRUE	FALSE
"2381_9"	FALSE	FALSE	FALSE	TRUE	FALSE
"7759_3"	FALSE	FALSE	FALSE	TRUE	FALSE
"3630_4"	FALSE	FALSE	FALSE	FALSE	FALSE
"9495_8"	FALSE	FALSE	FALSE	FALSE	TRUE
"8196_8"	FALSE	FALSE	TRUE	TRUE	TRUE

FeatureHashing

- `FeatureHashing` implements `split` in the formula interface.

```
hash_size <- 2^16
m.train <- hashed.model.matrix(~ split(review, delim = " ", type = "existence"),
                               data = imdb.train,
                               hash.size = hash_size,
                               signed.hash = FALSE)
m.valid <- hashed.model.matrix(~ split(review, delim = " ", type = "existence"),
                              data = imdb.valid,
                              hash.size = hash_size,
                              signed.hash = FALSE)
```

Type of `split`

- `existence`
- `count`
- `tf-idf` which is contributed by Michaël Benesty and will be announced in v0.9.1

Gradient Boosted Decision Tree with `xgboost`

```
dtrain <- xgb.DMatrix(m.train, label = imdb.train$sentiment)
dvalid <- xgb.DMatrix(m.valid, label = imdb.valid$sentiment)
watch <- list(train = dtrain, valid = dvalid)
g <- xgb.train(booster = "gblinear", nrounds = 100, eta = 0.0001, max.depth = 2,
               data = dtrain, objective = "binary:logistic",
               watchlist = watch, eval_metric = "auc")
```

```
[0]  train-auc:0.969895  valid-auc:0.914488
[1] train-auc:0.969982  valid-auc:0.914621
[2] train-auc:0.970069  valid-auc:0.914766
...
[97]  train-auc:0.975616    valid-auc:0.922895
[98]    train-auc:0.975658  valid-auc:0.922952
[99]    train-auc:0.975700  valid-auc:0.923014
```

Performance

- The AUC of g is 0.90110 in the public leader board
  - It outperforms the benchmark in Kaggle

The Purpose of FeatureHashing

- Make our life easier

Formula Interface in R

Algorithm in Text Book

Regression

$$y = X\beta + \varepsilon$$

Real Data

	SEPAL.LENGTH	SEPAL.WIDTH	PETAL.LENGTH	PETAL.WIDTH	SPECIES
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica

- How to convert the real data to X?

Feature Vectorization and Formula Interface

- X is usually constructed via `model.matrix` in R

```
model.matrix(~ ., iris.demo)
```

	(INTERCEPT)	SEPAL.LENGTH	SEPAL.WIDTH	PETAL.LENGTH	PETAL.WIDTH	SPECIESVERSICOLOR	SPECIESVIRGINI
1	1	5.1	3.5	1.4	0.2	0	0
2	1	4.9	3.0	1.4	0.2	0	0
51	1	7.0	3.2	4.7	1.4	1	0
52	1	6.4	3.2	4.5	1.5	1	0
101	1	6.3	3.3	6.0	2.5	0	1
102	1	5.8	2.7	5.1	1.9	0	1

Formula Interface

- `y ~ a + b`
  - `y` is the response
  - `a` and `b` are predictors

Formula Interface: +

- + is the operator of combining linear predictors

```
model.matrix(~ a + b, data.demo)
```

	(INTERCEPT)	A	B
1	1	5.1	3.5
2	1	4.9	3.0
51	1	7.0	3.2
52	1	6.4	3.2
101	1	6.3	3.3
102	1	5.8	2.7

Formula Interface: :

- : is the interaction operator

```
model.matrix(~ a + b + a:b, data.demo)
```

	(INTERCEPT)	A	B	A:B
1	1	5.1	3.5	17.85
2	1	4.9	3.0	14.70
51	1	7.0	3.2	22.40
52	1	6.4	3.2	20.48
101	1	6.3	3.3	20.79
102	1	5.8	2.7	15.66

Formula Interface: *

- * is the operator of cross product

```
# a + b + a:b
model.matrix(~ a * b, data.demo)
```

	(INTERCEPT)	A	B	A:B
1	1	5.1	3.5	17.85
2	1	4.9	3.0	14.70
51	1	7.0	3.2	22.40
52	1	6.4	3.2	20.48
101	1	6.3	3.3	20.79
102	1	5.8	2.7	15.66

Formula Interface: (

- : and * are distributive over +

```
# a:c + b:c
model.matrix(~ (a + b):c, data.demo)
```

	(INTERCEPT)	A:C	B:C
1	1	7.14	4.90
2	1	6.86	4.20
51	1	32.90	15.04
52	1	28.80	14.40
101	1	37.80	19.80
102	1	29.58	13.77

Formula Interface: .

- . means all columns of the `data`.

```
# ~ Sepal.Length + Sepal.Width + Petal.Length +
#   Petal.Width + Species
model.matrix(~ ., iris.demo)
```

	(INTERCEPT)	SEPAL.LENGTH	SEPAL.WIDTH	PETAL.LENGTH	PETAL.WIDTH	SPECIESVERSICOLOR	SPECIESVIRGINI
1	1	5.1	3.5	1.4	0.2	0	0
2	1	4.9	3.0	1.4	0.2	0	0
51	1	7.0	3.2	4.7	1.4	1	0
52	1	6.4	3.2	4.5	1.5	1	0
101	1	6.3	3.3	6.0	2.5	0	1
102	1	5.8	2.7	5.1	1.9	0	1

- Please check `?formula`

Categorical Features

Categorical Feature in R

- A categorical variables of K categories are transformed to a $K-1$-dimentional vector.
- There are many coding systems and the most commonly used is `Dummy Coding`.
  - The first category are transformed to $\vec{0}$.

Dummy Coding

```
##            versicolor virginica
## setosa              0         0
## versicolor          1         0
## virginica           0         1
```

Categorical Feature in Machine Learning

- Predictive analysis
- Regularization
- The categorical variables of K categories are transformed to K-dimentional vector.
  - The missing data are transformed to $\vec{0}$.

```
contr.treatment(levels(iris.demo$Species), contrasts = FALSE)
```

```
##            setosa versicolor virginica
## setosa          1          0         0
## versicolor      0          1         0
## virginica       0          0         1
```

Motivation of the FeatureHashing

Kaggle: Display Advertising Challenge

- Given a user and the page he is visiting, what is the probability that he will click on a given ad?

The Size of the Dataset

- 13 integer features and 26 categorical features
- 7×10^7 instances
- Download the dataset via this link

Vectorize These Features in R

- After binning the integer features, I got 3×10^7 categories.
- Sparse matrix was required
  - A dense matrix required 14PB memory...
  - A sparse matrix required 40GB memory...

Estimating Computing Resources

- Dense matrix: `nrow` $(7 \times 10^7) \times$ `ncol` $(3 \times 10^7) \times 8$ bytes
- Sparse matrix: `nrow` $(7 \times 10^7) \times (13 + 26) \times (4 + 4 + 8)$ bytes

Vectorize These Features in R

- `sparse.model.matrix` is similar to `model.matrix` but returns a sparse matrix.

Fit Model with Limited Memory

- A post Beat the benchmark with less then 200MB of memory describes how to fit a model with limited resources.
  - online logistic regression
  - feature hash trick
  - adaptive learning rate

Online Machine Learning

- In online learning, the model parameter is updated after the arrival of every new datapoint.
  - Only the aggregated information and the incoming datapoint are used.
- In batch learning, the model paramter is updated after access to the entire dataset.

Memory Requirement of Online Learning

- Related to model parameters
  - Require little memory for large amount of instances.

Limitation of `model.matrix`

- The vectorization requires all categories.

```
contr.treatment(levels(iris$Species))
```

```
##            versicolor virginica
## setosa              0         0
## versicolor          1         0
## virginica           0         1
```

My Work Around

- Scan all data to retrieve all categories
- Vectorize features in an online fashion
- The overhead of exploring features increases

Observations

- Mapping features to $\{0, 1, 2, \ldots, K\}$ is one of method to vectorize feature.
  - `setosa` => $\vec{e}_1$
  - `versicolor` => $\vec{e}_2$
  - `virginica` => $\vec{e}_3$

```
##            setosa versicolor virginica
## setosa          1          0         0
## versicolor      0          1         0
## virginica       0          0         1
```

Observations

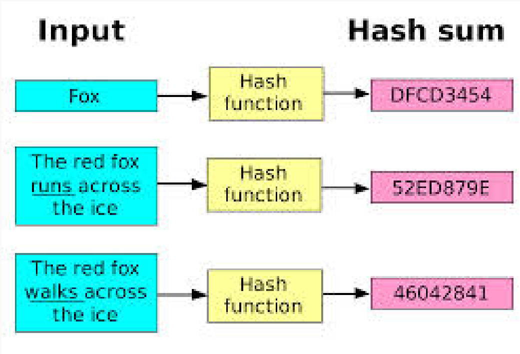
- `contr.treatment` ranks all categories to map the feature to integer.
- What if we do not know all categories?
    - Digital features are integer.
    - We use a function maps $\mathbb{Z}$ to $\{0, 1, 2, \ldots, K\}$

```
charToRaw("setosa")
```

```
## [1] 73 65 74 6f 73 61
```

What is Feature Hashing?

- A method to do feature vectorization with a hash function



- For example, Mod `%%` is a family of hash function.

Feature Hashing

- Choose a hash function and use it to hash all the categorical features.
- The hash function does not require global information.

An Example of Feature Hashing of Criteo's Data

V15	V16	V17
68fd1e64	80e26c9b	fb936136
68fd1e64	f0cf0024	6f67f7e5
287e684f	0a519c5c	02cf9876
68fd1e64	2c16a946	a9a87e68
8cf07265	ae46a29d	c81688bb
05db9164	6c9c9cf3	2730ec9c

- The categorical variables have been hashed onto 32 bits for anonymization purposes.
- Let us use the last 4 bits as the hash result, i.e. the hash function is `function(x) x %% 16`
- The size of the vector is (2^4), which is called `hash size`

An Example of Feature Hashing of Criteo's Data

V15	V16	V17
68fd1e64	80e26c9b	fb936136
68fd1e64	f0cf0024	6f67f7e5
287e684f	0a519c5c	02cf9876
68fd1e64	2c16a946	a9a87e68
8cf07265	ae46a29d	c81688bb
05db9164	6c9c9cf3	2730ec9c

- 68fd1e64, 80e26c9b, fb936136 => 4, b, 6
  - (0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
- 68fd1e64, f0cf0024, 6f67f7e5 => 4, 4, 5
  - (0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Hash Collision

- `68fd1e64, f0cf0024, 6f67f7e5` => `4, 4, 5`
  - (0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
- Hash function is a many to one mapping, so different features might be mapped to the same index. This is called collision.
- In the perspective of statistics, the collision in hash function makes the effect of these features confounding.

Choosing Hash Function

- Less collision rate
  - Real features, such as text features, are not uniformly distributed in $\mathbb{Z}$.
- High throughput
- `FeatureHashing` uses the Murmurhash3 algorithm implemented by `digest`

Pros of FeatureHashing

A Good Companion of Online Algorithm

```
library(FeatureHashing)
hash_size <- 2^16
w <- numeric(hash_size)
for(i in 1:1000) {
  data <- fread(paste0("criteo", i))
  X <- hashed.model.matrix(V1 ~ ., data, hash.size = hash_size)
  y <- data$V1
  update_w(w, X, y)
}
```

Pros of FeatureHashing

A Good Companion of Distributed Algorithm

```
library(pbdMPI)
library(FeatureHashing)
hash_size <- 2^16
w <- numeric(hash_size)
i <- comm.rank()
data <- fread(paste0("criteo", i))
X <- hashed.model.matrix(V1 ~ ., data, hash.size = hash_size)
y <- data$V1
# ...
```

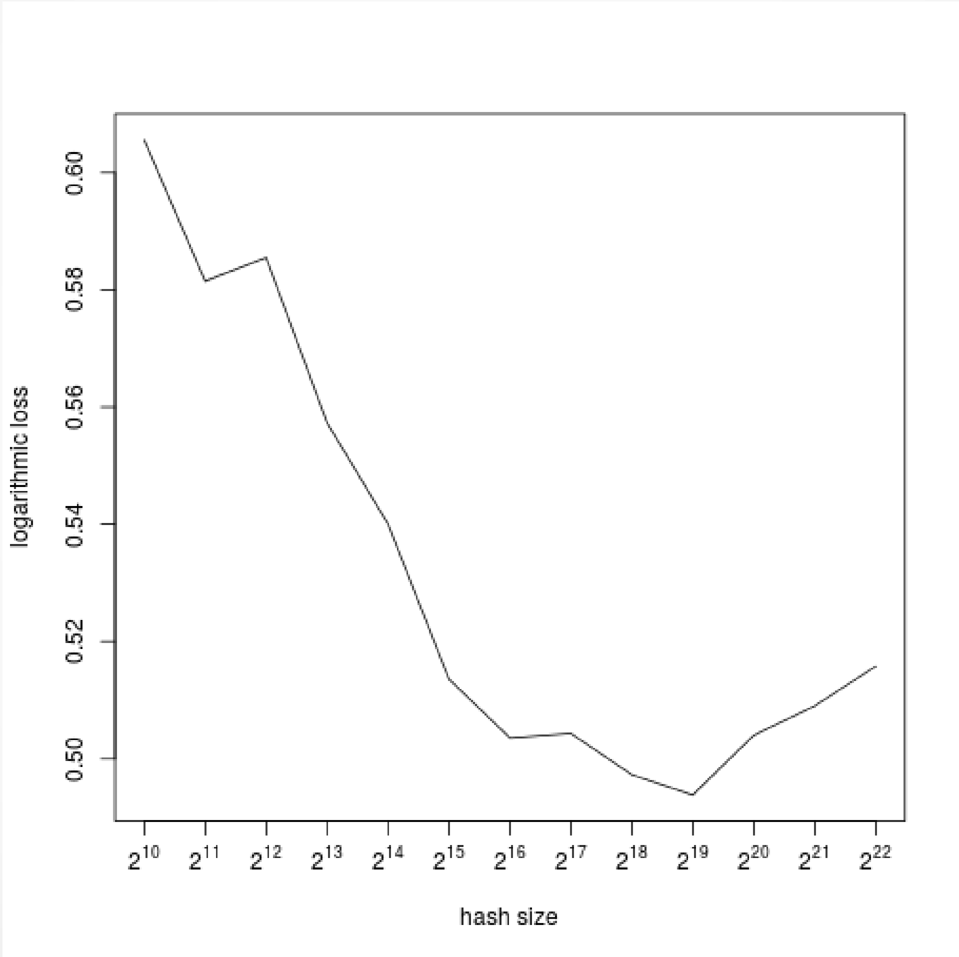
Pros of FeatureHashing

Simple Training and Testing

```
library(FeatureHashing)
model <- is_click ~ ad * (url + ip)
m_train <- hashed.model.matrix(model, data_train, hash_size)
m_test <- hashed.model.matrix(model, data_test, hash_size)
```

Cons of FeatureHashing

Hash Size



Cons of FeatureHashing

Lose Interpretation

- Collision makes the interpretation harder.
- It is inconvenient to reverse the indices to feature.

```
m <- hashed.model.matrix(~ Species, iris, hash.size = 2^4, create.mapping = TRUE)
hash.mapping(m) %% 2^4
```

```
##     Speciessetosa  Speciesvirginica Speciesversicolor
##                 7                13                 8
```

The Result of the Competition...

- No.1: Field-aware Factorization Machine
  - The hashing trick do not contribute any improvement on the leaderboard. They apply the hashing trick only because it makes their life easier to generate features..
- No.3, no.4 and no.9 (we): Neuron Network

Extending Formula Interface in R

- Formula is the most R style interface

Tips

- `terms.formula` and its argument `specials`

```
tf <- terms.formula(~ Plant * Type + conc * split(Treatment), specials = "split",
                    data = CO2)
attr(tf, "factors")
```

```
##                  Plant Type conc split(Treatment) Plant:Type
## Plant                1    0    0                0          1
## Type                 0    1    0                0          1
## conc                 0    0    1                0          0
## split(Treatment)     0    0    0                1          0
##                  conc:split(Treatment)
## Plant                                0
## Type                                 0
## conc                                 1
## split(Treatment)                     1
```

Specials

- `attr(tf, "specials")` tells which rows of `attr(tf, "factors")` need to be parsed further

```
rownames(attr(tf, "factors"))
```

```
## [1] "Plant"            "Type"             "conc"
## [4] "split(Treatment)"
```

```
attr(tf, "specials")
```

```
## $split
## [1] 4
```

Parse

- `parse` extracts the information from the `specials`

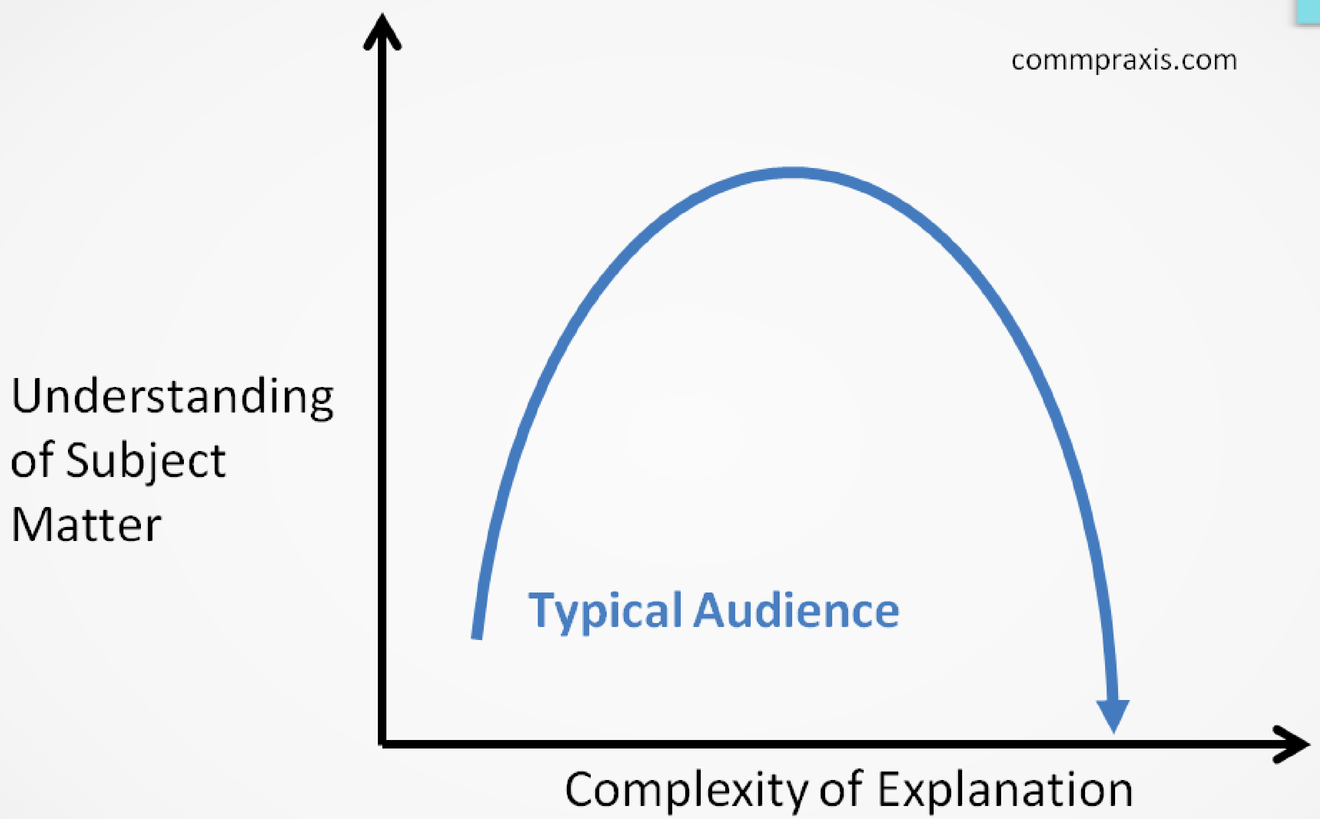
```
options(keep.source = TRUE)
p <- parse(text = rownames(attr(tf, "factors"))[4])
getParseData(p)
```

```
##   line1 col1 line2 col2 id parent                token terminal      text
## 9     1    1     1   16  9      0                 expr    FALSE          
## 1     1    1     1    5  1      3 SYMBOL_FUNCTION_CALL     TRUE     split
## 3     1    1     1    5  3      9                 expr    FALSE          
## 2     1    6     1    6  2      9                  '('     TRUE         (
## 4     1    7     1   15  4      6               SYMBOL     TRUE Treatment
## 6     1    7     1   15  6      9                 expr    FALSE          
## 5     1   16     1   16  5      9                  ')'     TRUE         )
```

Efficiency

- `Rcpp`
  - The core functions are implemented in C++
- Invoking external C functions
  - `digest` exposes the C function:
    - Register the c function
    - Add the helper header file
  - `FeatureHashing` imports the C function:
    - Import the c function

Let's Discuss the Implementation Later



source: http://www.commpraxis.com/wp-content/uploads/2015/01/commpraxis-audience-confusion.png

Summary

- Pros of `FeatureHashing`
  - Make it easier to do feature vectorization for predictive analysis.
  - Make it easier to tokenize the text data.
- Cons of `FeatureHashing`
  - Decrease the prediction accuracy if the `hash size` is too small
  - Interpretation becomes harder.

When should I use `FeatureHashing`?

Short Answer: Predictive Anlysis with a large amount of categories.