+ Add article

Text Analytics

View
Edit
Changes

LAST MODIFIED:
July 9, 2015, 11:25 a.m.

See all children

Text analytics is a set of techniques that model and structure the information content of textual sources, which are frequently loosely structured and complex. The ultimate goal is to convert text into data for analysis.

One popular and commonly-used text analytics technique is called "bag of words". While fully understanding text is difficult, this approach does something much simpler: it just counts the number of times each word appears in the text. So ultimately, you end up with one feature for each word.

This approach is often very effective, and it can be dramatically improved by pre-processing the text. Text data has many inconsistencies that can cause algorithms trouble. Some common initial pre-processing steps are to convert all of the letters to lowercase and to remove punctuation. This makes sure that "analytics", "AnALYticS", "Analytics!", and "#analytics" are all considered the same word.

More advanced pre-processing techniques include the removal of stop words and stemming the words. Stop words are words like "the", "at", "is", and "which". These words are frequently used, but are not typically meaningful in a particular sentence. Since they are unlikely to improve the quality of an analytics model, they are removed to clean up the dataset. Stemming words means representing different forms of the same words by a common stem. An example is representing "argue", "argued", "argues" and "arguing" by the common stem "argu". It typically does not matter which form of the word is used, so by combining words with common stems analytics models can often be improved. There are several ways to approach the problem of stemming, including building a database of words and their stems, and writing a rule-based algorithm.

Text Analytics in R

To use text analytics in R, we'll use the package "tm". For other packages see this page on CRAN. Suppose your dataset is called "DataFrame", and the text field for which you would like to use the bag of words approach is called "Text".

First, we need to create a corpus of our text field:

```
TextCorpus = Corpus(VectorSource(DataFrame$Text))
```

Then, we need to pre-process the text. We'll convert all letters to lowercase, remove punctuation, remove stopwords, and stem the words. The second line of code is due to a recent change in the "tm" package, and needs to be run if you are using the latest version of tm.

```
TextCorpus = tm_map(TextCorpus, tolower)
TextCorpus = tm_map(TextCorpus, PlainTextDocument)
TextCorpus = tm_map(TextCorpus, removePunctuation)
TextCorpus = tm_map(TextCorpus, removeWords, stopwords("english"))
TextCorpus = tm_map(TextCorpus, stemDocument, language="english")
```

Now, we will create the bag of words matrix of frequencies:

```
FrequencyMatrix = DocumentTermMatrix(TextCorpus)
```

If desired, we can create a sparse matrix, by removing words that occur very rarely:

```
SparseMatrix = removeSparseTerms(FrequencyMatrix, 0.95)
SparseMatrix = removeSparseTerms(FrequencyMatrix, 0.99)
```

The first line only keeps words that appear in 5% or more of the observations, and the second line keeps words that appear in 1% or more of the observations (the closer the second argument is to 1, the more words will be in the resulting matrix).

Lastly, we can convert our matrix to a data frame, and make sure all of the variable names (words) are "R-friendly":

```
NewDataFrame = as.data.frame(as.matrix(SparseMatrix))
colnames(NewDataFrame) = make.names(colnames(NewDataFrame))
```

Now, you can add a dependent variable back into this data frame to build a predictive model using the words in the text.

Interesting info from the last (optional) homework in the Text Analytics module:

USING N-GRAMS

Another source of information that might be extracted from text is the frequency of various n-grams. An n-gram is a sequence of n consecutive words in the document. For instance, for the document "Text analytics rocks!", which we would preprocess to "text analyt rock", the 1-grams are "text", "analyt", and "rock", the 2-grams are "text analyt" and "analyt rock", and the only 3-gram is "text analyt rock". n-grams are order-specific, meaning the 2-grams "text analyt" and "analyt text" are considered two separate n-grams. We can see that so far our analysis has been extracting only 1-grams.

We do not have exercises in this class covering n-grams, but if you are interested in learning more, the "RTextTools", "tau", "RWeka", and "textcat" packages in R are all good resources.

Corpus Preprocessing function

```
tm_pre_process <- function(data){
        library(tm)
        corpusTitle = Corpus(VectorSource(data))
        corpusTitle = tm_map(corpusTitle, tolower)
        corpusTitle = tm_map(corpusTitle, PlainTextDocument)
        corpusTitle = tm_map(corpusTitle, removePunctuation)
        corpusTitle = tm_map(corpusTitle, removeWords, stopwords("english"))
        corpusTitle = tm_map(corpusTitle, stemDocument)
        corpusTitle
}
```

Function can be loaded directly into R. Then can be re used for doing repetitive pre-processing.

For an example

```
corpus = tm_pre_process(daata)
```