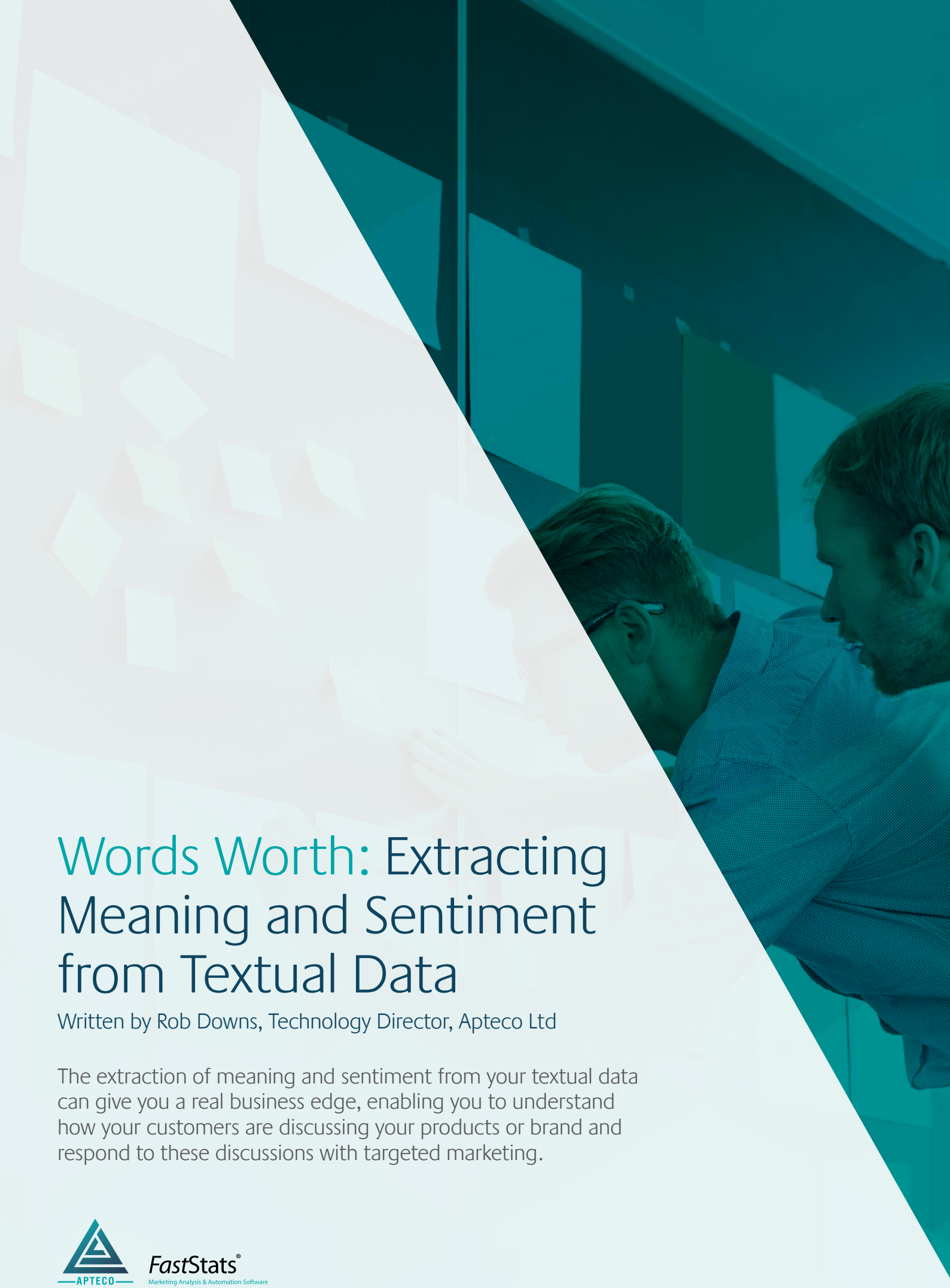


# Words Worth: Extracting Meaning and Sentiment from Textual Data

Written by Rob Downs, Technology Director, Apteco Ltd

The extraction of meaning and sentiment from your textual data can give you a real business edge, enabling you to understand how your customers are discussing your products or brand and respond to these discussions with targeted marketing.

APTECO

FastStats®
Marketing Analysis & Automation Software

# Extracting Meaning and Sentiment from Textual Data

Through posted comments, tweets and reviews your customers can provide you with a lot of useful feedback. It has become essential to rapidly glean information from textual data in order to respond to your market by adapting your products and marketing message.

FastStats includes powerful tools for analysing textual data and we are working on extending the capabilities of these tools with the goal of modelling sentiment.

In a previous blog post 'They Said What? Textual data analysis approaches' I described five ways in which you could analyse textual data. Here I will put some of these methods to work in FastStats using a set of real and not particularly easy data.

## Good and Bad Reviews

Ideally we want a model that tells us whether a review is good or bad. This will be difficult as language is complex but let's see how far we can get.

To explore these ideas I've used a dataset of movie reviews. The data is sourced from IMDB and made available for data analysis research by Andrew L. Maas et. al

Fifty thousand films are reviewed. FastStats has a limit of 255 characters for text so I have built a FastStats system that has a transactional record for each line of text in the review. Helpfully the reviews are also marked with a rating from 1-9, the higher the rating the better received the film.

So a typical review looks like this:

| URN | Rating | Line | Text |
|---|---|---|---|
| 1 | 2 | 1 | Once again Mr. Costner has dragged out a movie for far longer than necessary. Aside from the terrific |
| 1 | 2 | 2 | sea rescue sequences, of which there are very few I just did not care about any of the characters. Most |
| 1 | 2 | 3 | of us have ghosts in the closet, and Costner's character are realized early on, and then forgotten until |
| 1 | 2 | 4 | much later, by which time I did not care. The character we should really care about is a very cocky, |
| 1 | 2 | 5 | overconfident Ashton Kutcher. The problem is he comes off as kid who thinks he's better than anyone else |
| 1 | 2 | 6 | around him and shows no signs of a cluttered closet. His only obstacle appears to be winning over Costner |
| 1 | 2 | 7 | Finally when we are well past the half way point of this stinker, Costner tells us all about Kutcher's |
| 1 | 2 | 8 | ghosts. We are told why Kutcher is driven to be the best with no prior inkling or foreshadowing. No magic |
| 1 | 2 | 9 | here, it was all I could do to keep from turning it off an hour in. |

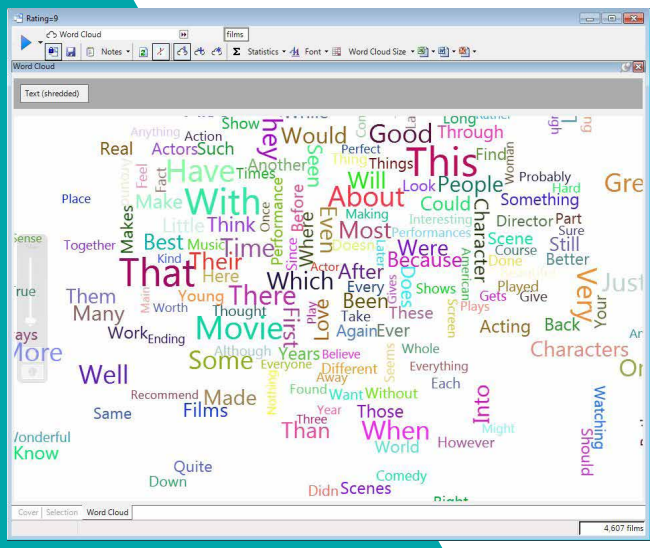
With a rating of only 2 this is a negative review and I could detect that tone by noticing words like “dragged”, “stinker”, “overconfident”. However it also includes “winning”, “terrific” and “better” which appear positive when taken out of context.

Shorter more succinct text would be easier to interpret. Tweets also contain tags which give obvious reference points. So, I’ve chosen a relatively hard dataset.

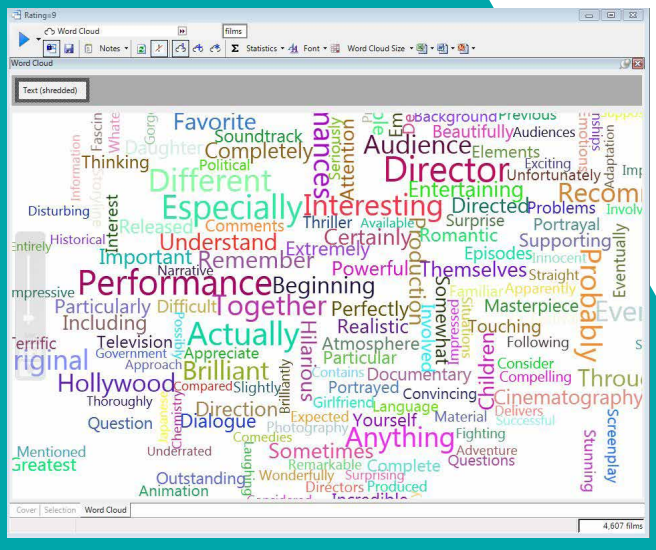
# Word Cloud

First I will try a word cloud with the review line text analysed as a shredded dimension. This breaks the reviews up into words and highlights their relative frequencies of use. Here I’ve selected just the reviews of films with the top rating 9.

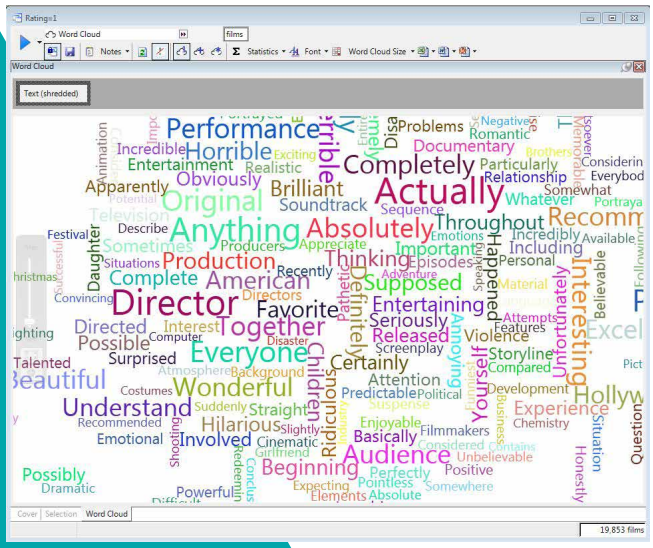
This isn’t as useful as I’d hoped: it is just highlighting common words. By changing the “shredding” properties I can filter out shorter words.



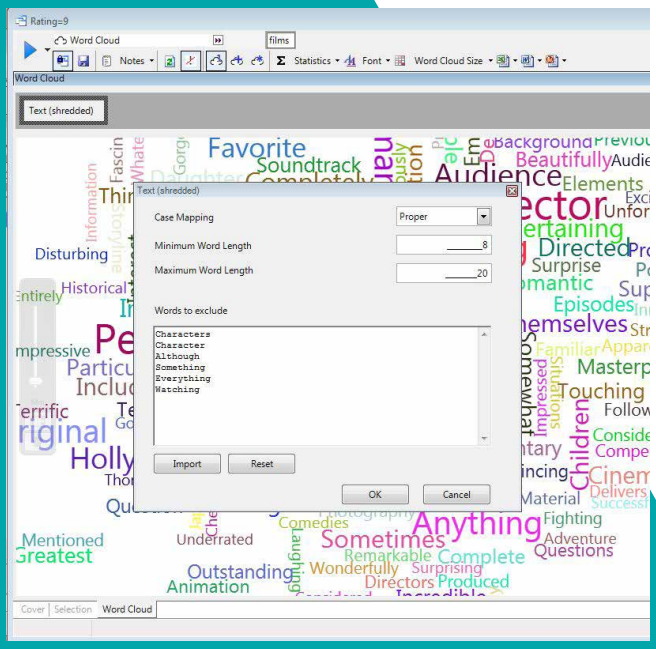

This looks more promising as I can see some positive words coming to the fore. But actually we’re still seeing a lot of common words.



When I change the selection to instead be the worst films (rating 1) the resulting cloud is rather similar until we peek into the detail. Words like “actually” and “director” are likely to occur in both good and bad reviews.




I can use the shredded properties to exclude common words.



And by doing the same analysis on a tree I could get the list of those common words. However some of these common words are going to be significant.

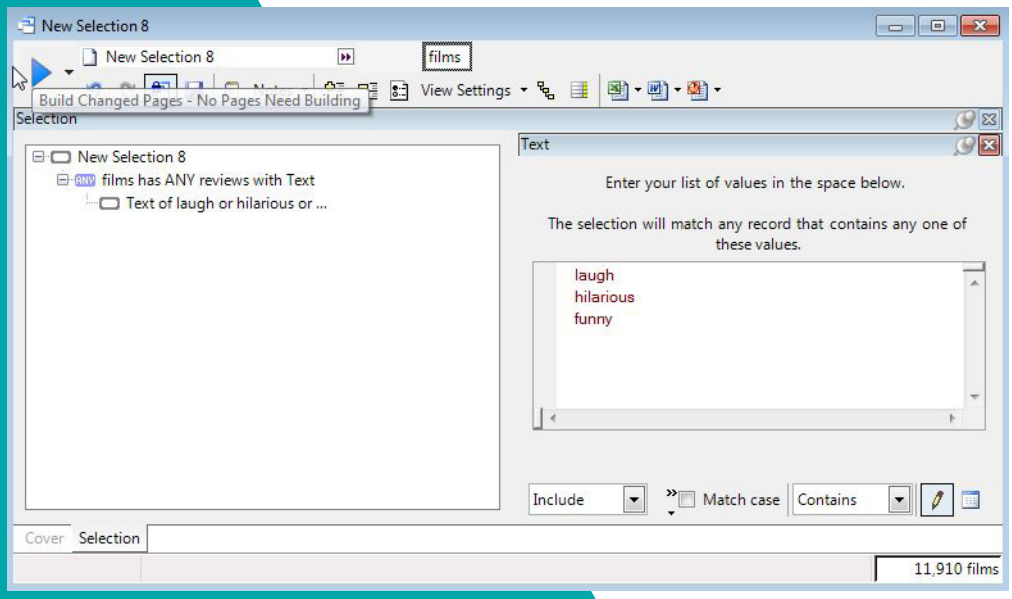
So although this approach gives me some nice visuals it isn't going to get much further with separating the good and bad movie reviews.



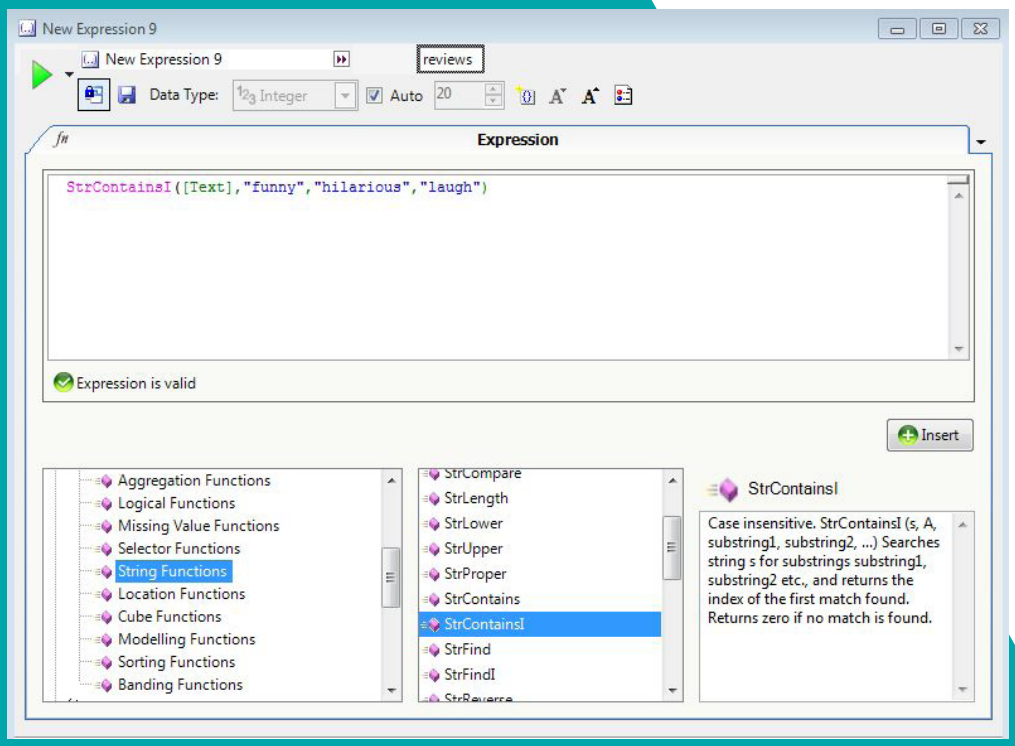
# Text Selection

We can of course select directly on words or phrases by using the text selection or text expressions. This is useful if we want to pick out all reviews that contain a particular hashtag, phrase or words.

Here I've selected films with reviews that include the words "laugh", "hilarious" or "funny".




I could also do this selection using a string function in an expression. This is a more flexible option if you need to manipulate the text or if the target words are variable.



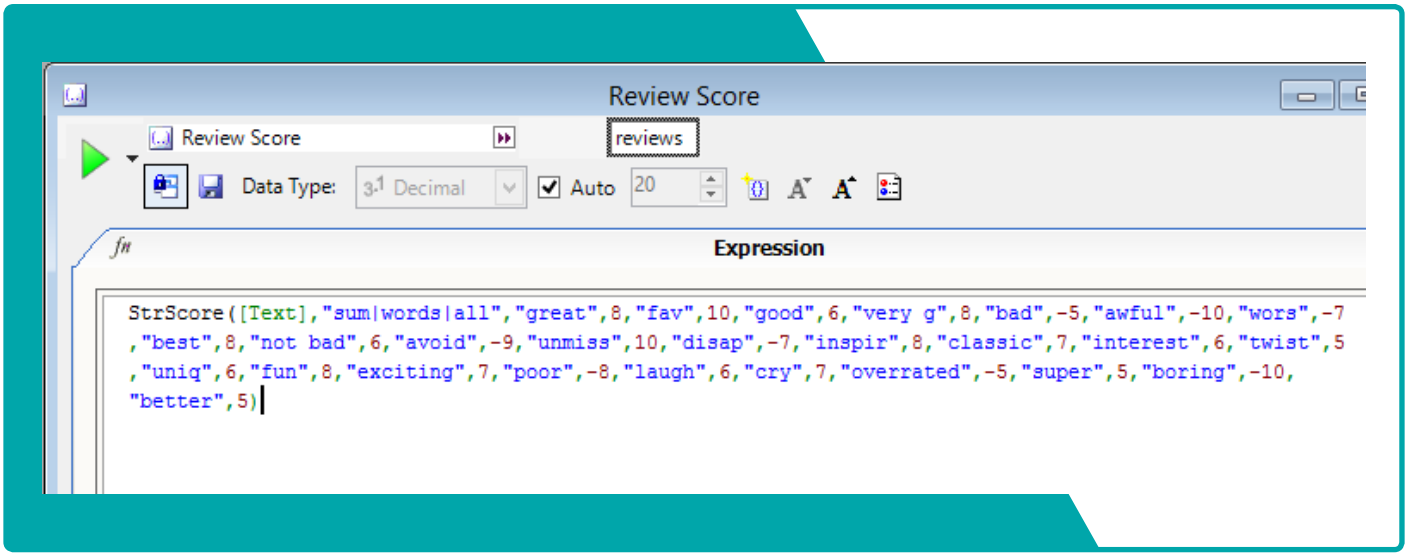

Text selection is often a good start. A train operating company might gain a lot of insight on late running trains and customer mood by searching for the words "delay" and "delayed" in tweets.

# Scoring Words

Going back to the movie reviews we recognise that there are some positive and negative words. So I could assign a score say between +10 for words associated with a good review down to -10 for those associated with a bad review.

Then for each review I can add up (or maybe average) the points to give an overall score.

The new score expression function (StrScore) does exactly this by searching for each substring, assigning and aggregating the scores.




So with this formula a review that reads:

## “great film with a uniquely funny twist”

scores 8 + 6 + 8 + 5 = 27

while

## “overrated - a few laughs but so boring”

scores -5 +6 -10 = -9

This function can be used for a basic sentiment analysis. Obviously it is limited to the words supplied and is not going to get it quite right on phrases like “awfully good”, “laughably poor” - unless you specifically add these.

This method is quite effective for short text using concise language where we know the words that are likely to be important. For example short reviews or tweets referencing a limited product range, a new model of phone or car for example.

In the example above I’ve just included words and scores based on intuition. To cope with longer, more varied text I really need some method of getting these from the data.

# Good Words, Bad Words

With the movie review data I know the overall rating so can I use this to find the words most associated with good and bad reviews?

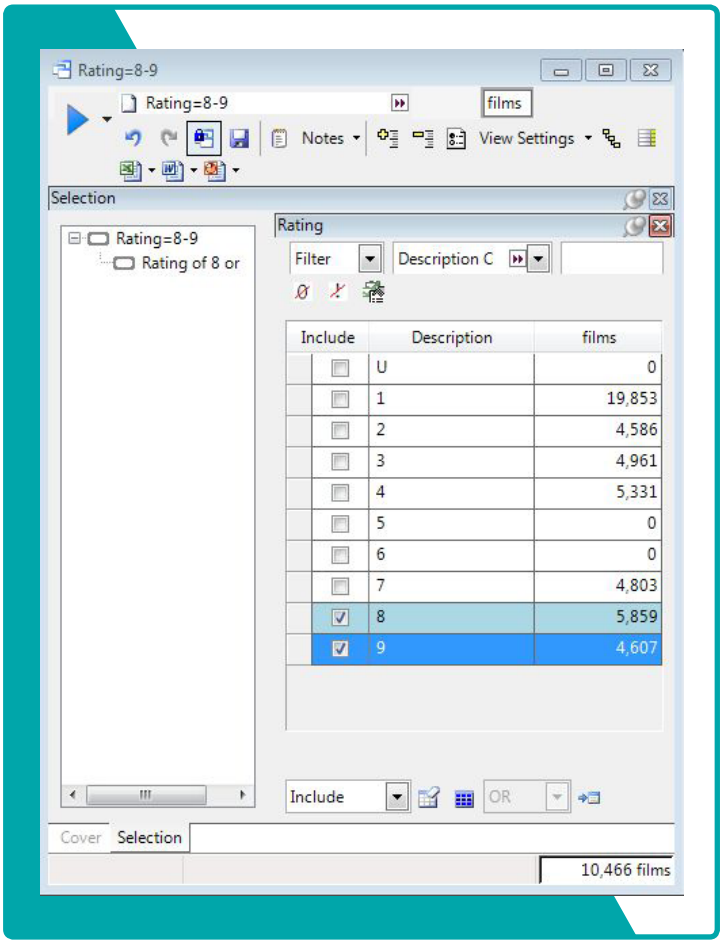
I'll first select exactly 10,000 films that have a rating of 1 (worst) and 10,000 that have a rating of 8 or 9 (best).

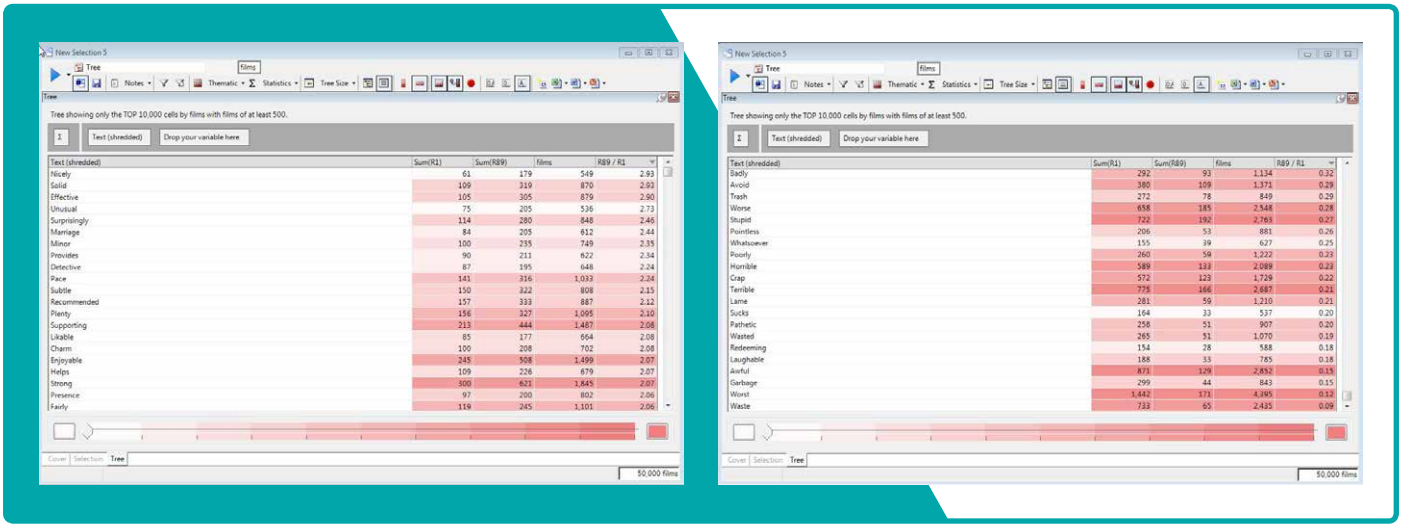
I can then use a tree to analyse words occurring in these two selections and across all films. The first column Sum(R1) shows the number of occurrences of the word in films rated 1, the second column Sum(89) shows the number of occurrences of the word in films rated 8 or 9.

I have limited the tree to only show words that occur at least 500 times across all reviews.

So from the top line I can see that the word "Nicely" occurs 61 times in bad reviews, 179 times in good reviews. Because I've used equally sized selections for R1 and R89 I can divide these totals directly to give an "odds ratio". In this case the odds of a good review are nearly tripled if the word "nicely" is detected (179/61 = 2.93).

The tree has been sorted by this ratio and jumping to the bottom page of the results we see words like "garbage", "worst" and "waste". Waste has an odds ratio of 0.09 which I can turn round (1/0.09 = 11.11) to imply that chances of a review being bad are increased by a factor of over 11 if it includes the word "waste".



# Building A Text Model

I can turn these results into a predictive model for good/bad reviews. The log of the odds ratios are the coefficients for a logistic regression.

I extended the tree output to include a text model section that specifies these coefficients and saves them to a file.

```
<TextModel intercept="0.0">
<Text word="This" coefficient="-0.102382" />
<Text word="Waste" coefficient="-2.422758" />
<Text word="Chilling" coefficient="0.730888" />
<Text word="Confusion" coefficient="0.432864" />
<Text word="Delight" coefficient="0.659246" />
<Text word="Host" coefficient="0.302281" />
<Text word="Succeed" coefficient="0.339507" />
<Text word="Bitter" coefficient="0.589963" />
<Text word="Treasure" coefficient="0.263585" />
<Text word="Idiotic" coefficient="-1.425009" />
<Text word="Upset" coefficient="-0.282567" />
<Text word="Lazy" coefficient="-0.280302" />
<Text word="Seasons" coefficient="-0.013072" />
<Text word="Convey" coefficient="0.205444" />
<Text word="Struggles" coefficient="0.662688" />
<Text word="Cynical" coefficient="0.444251" />
<Text word="Embarrassed" coefficient="-1.203973" />
<Text word="Guessing" coefficient="0.253196" />
<Text word="Happiness" coefficient="0.551497" />
```

This text model gives us the list of words and scores like those that I previously guessed in order to use the StrScore function.

The intercept would be based on the overall odds of getting a good or bad review and is easily calculated. There are actually far more bad reviews than good reviews so for accurate prediction this would be required. For our purposes of scoring a new review it is more useful to omit this effect.

I then extended the StrScore function so that it can read the text model from a file.

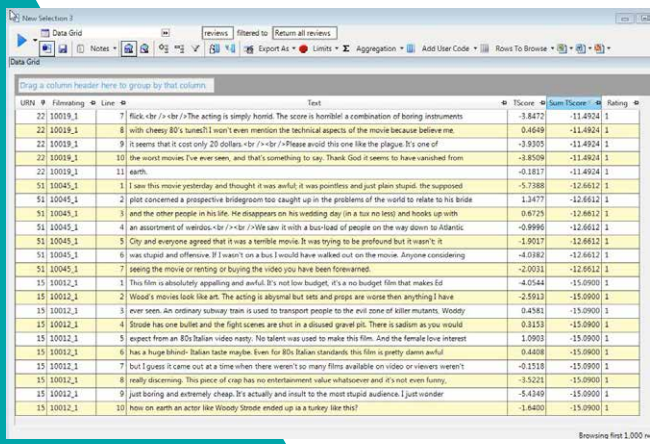
StrScore([Text],"sum","C:\FastStats\imdb\save\textmodel.xml")

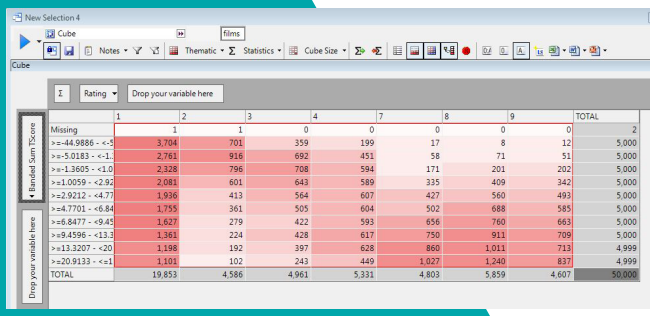
I ran this to calculate scores for all review lines. Using the aggregation wizard I summed the score across all the review lines to give an overall score for the film.

Some example results for a couple of poorly rated films are shown below.

Analysing the banded film review score against the actual rating shows that the higher scores are associated with the better films and vice versa. So the model is successful up to a point although as expected it is far from perfect.

This review data is actually particularly hard to classify as it is quite common to reference other better or worse films within a review. Product reviews within a particular sector or brand would probably be easier and get better modelled results.





# Improving the Text Model

By studying the film reviews that are wrongly classified it is easy to see that a perfect model would require a deep understanding of the linguistics, context and a human feel for humour and sarcasm.

One obvious improvement would be to consider the effects of negating words like "not", "no", "never" which often reverse the meaning and hence negate the coefficient. Conversely words like "very", "really", "absolutely" could be detected and used to magnify the coefficient of the following word.

However these enhancements would need to take account of word order in the language whereas the basic model above is language neutral.

I've added support for pre-scored "qualifier" words in the TextModel:

```
<Text qualifier="not" coefficient="-1.0" />
<Text qualifier ="very" coefficient="1.5" />
<Text qualifier ="never" coefficient="-1.2" />
<Text qualifier ="exceptionally" coefficient="2.0" />
```

# A Last Word

The TextModel and StrScore functionality that I've used in this analysis is a result of our R&D. We are planning to integrate these features into future versions of FastStats Discoverer.

If you have a large amount of interesting textual data and / or have a requirement to analyse reviews, tweets, comments etc. then we'd like to hear from you.

# Takeaways:



Textual analysis gives you insight into the discussions consumers are having around your brand.



Analysis of the sentiment surrounding this discussion can drive your marketing campaigns.

# About Apteco

Apteco Ltd is a privately owned software development business specialising in marketing analysis and automation solutions. Originally established in 1987, we continue to invest heavily in the next generation of FastStats®.

Our team of dedicated developers, support consultants and project managers continues to grow to support our international partner network of marketing service providers in the UK, Europe, North America and Australia. There are now over 400 FastStats® systems worldwide and over 3,500 users, including 500 campaign management users across 100 systems.

Apteco Ltd headquarters are in Warwick, UK and we also have an office in Frankfurt, Germany.


www.apteco.com

info@apteco.com

+44 (0)1926 407565


APTECO

FastStats®

Marketing Analysis & Automation Software