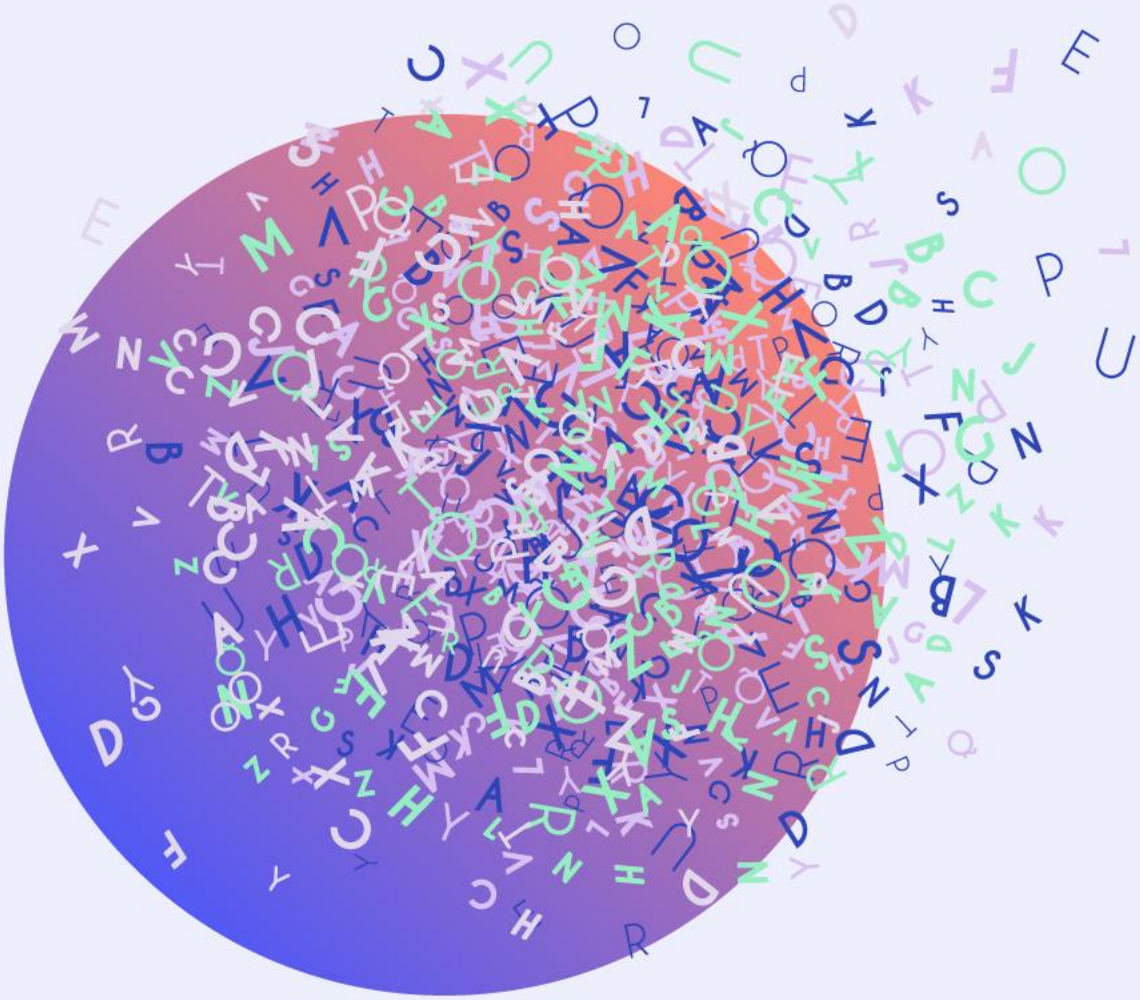
DISPLAYR

Automated Text Coding



Analysis and reporting software

displayr.com

052024

Table of Contents

Executive summary

This white paper presents a case study illustrating how automated categorization can be used to dramatically improve the productivity of categorizing text data, compared to manual categorization (i.e., “coding”).

The case study shows that the gain in productivity was at least 134%, and arguably much higher.

The resulting categorization has accuracy at a level that would be expected if a subject matter expert had manually performed the categorization.

This study should not be regarded as being representative. Some data sets are inevitably more suited to automatic categorization than others, and such decisions should be made on a case-by-case basis. This white paper illustrates how to make such a decision.

Automated text analysis in market research

There are three broad approaches to automating text analysis in market research:

1. Fully automated text analysis, where a table or visualization is automatically generated that summarizes the text.
2. Human-curated algorithms, where a data scientist creates bespoke code for each data set.
3. Predictive modeling, where a human manually categorizes a subset of the data, and machine learning is used to categorize the rest of the data.

Traditionally, market researchers have analyzed text data by reading through all the text data, creating a set of categories that are thought to accurately capture the variety of responses, and assign each text response into one or more of the categories. This process is known in market research as *coding.*

Coding performed by people is an extremely expensive way of summarizing text data, as the cost is proportional to the amount of data to be coded. As a result, market researchers have long been keenly interested in automated text analysis solutions. This section of the white paper provides a brief overview of the three approaches used to automate text analysis, describing why the third approach – predictive modeling of manually coded data – is the gold standard.

Fully automated text analysis

It is easy to articulate the ideal outcome of machine learning with text data: the user clicks a button and an insightful summary of the text is provided.

Modern text analysis software contains tools that do precisely this. The main variants of fully automated text analysis are:

- Data visualizations, such as word clouds and word maps. For example, in Q and Displayr these are provided in **Text Analysis > Word Clouds** and **Text Analysis > Advanced > Map**.
- Automatic phrase detection algorithms, which identify words and phrases that occur commonly, taking into account spelling errors and synonyms: **Text Analysis > Advanced > Setup Text Analysis** and **Text Analysis > Automatic Categorization > List of Items**.
- Entity extraction algorithms, that look for known named entity types. These pre-defined categories include real-world objects such as people, locations and organizations; temporal and numeric expressions such as dates, money, and other numeric measures; and abstract concepts such as religion, ideology, and criminal charge: **Text Analysis > Automatic Categorization > Entity Extraction**.
- Text clustering algorithms, that group people based on the similarity of the meaning of their text: **Text Analysis > Automatic Categorization > Unstructured Text**.

While such algorithms can be useful when in a rush, they are always quite error-prone, due to the sheer difficulty of the problem of accurately summarizing text data.

A simple example illustrates the basic problem. A survey question asked what people *disliked* about their phone company. One response to this question was:

> *"Nothing. I love them and would recommend them to anyone. The service was amazing and affordable"*

To a human, it is obvious that the respondent dislikes nothing about their phone company. But, all fully automated text analysis algorithms find such data problematic. The most widely used automated algorithms, such as word clouds, will incorrectly count this data towards the totals of people listing *service* and *price* as reasons for disliking, which is precisely the opposite of what it means.

Human curated algorithms

One way of improving automated text analyses is to have humans create new or modify existing algorithms. At a simplistic level, this can involve having users use their judgment to create rules. In the example described in the previous section, a rule such as AUTOMATICALLY CATEGORIZE AS *nothing* IF “nothing” APPEARS IN THE TEXT would likely be adequate.

In practice, human curation of algorithms has a number of challenges:

- It requires a high level of expertise. Typically, it involves writing code in R and Python.
- The only way to check the quality of the algorithms is to read through the text, so it can end up taking even more time than human coding. In the example of “nothing”, it is only by reading such a response that you can work out that you need to create the rule.
- It can be hard to check. The only way to check that it is sufficiently accurate is to compare it to manual coding of the data. But, if performing manual coding then it is typically the case that a better outcome is to use machine learning to predict the manual coding, as described next.

Predictive modeling

Predictive modeling is used to automate text analysis as follows:

- A subject matter expert manually categorizes some of the text data. For example, a random selection of 600 responses from a database of 100,000.
- The machine-learning algorithm uses the 600 manually categorized responses to predict the categorization of the remaining 99,400 uncategorized responses.
- To estimate the performance of this predictive approach, the same machine learning algorithm is performed solely using the manually categorized responses whereby some of the manually categorized responses are hidden by the algorithm for test purposes. For example, the machine learning algorithm is trained on 400 responses. The performance of the

algorithm is then checked by assessing its accuracy at predicting the 200 responses that were categorized but not used in the training.

This is the approach that is the focus of this working paper.

Methodology

> This section of the case study provides an overview of the data and methodology used in this white paper.

The study

The data set used in this test was collected in mid-2019 by Qualtrics from a representative study of cell phone users. The total sample size was 2,090. The data was collected over 12 weeks.

Respondents were asked:

*What do you like about **INSERT PHONE COMPANY** as a cell phone provider? Type "Nothing" if there is nothing that you like.*

The first 20 responses from the question are shown in the table below. Two key features are evident in the data:

- The text responses are comparatively short. This is, in general, desirable if using automated text analysis. The longer the data, the more subjectivity that tends to be required to interpret it.
- A *code frame* is required that permits a respondent's data to be in multiple categories (i.e., *multiple response coding*). For example, respondent three makes two distinct points, and respondent nine makes two or three.

	Text	Likes
1	great offers
2	Discount
3	No contracts and ability to set up payment arrangements
4	I like that they are reliable.
5	it is well
6	everything
7	Nothing
8	nothing
9	the rates are reasonable. i like tmobile tuesdays. i feel appreciated by the company.
10	no contract and low rates
11	Nothing
12	they have good customer service
13	I like my plan, which is unlimited talk and data, and I like that the network is very stable.
14	the price
15	That I have good reception and never have billing problems
16	They have terrific customer service and I get a good deal with a bundle.
17	Financing
18	Speed
19	very likely
20	Everything

Is the study representative?

This study should not be regarded as being representative. Some data sets are inevitably more suited to automatic categorization than others, and such decisions should be made on a case-by-case basis.

The coder

The coding was performed by a market researcher with more than 20 years' experience and extensive experience in the cell phone market, as both a consultant and the insights director of a telco.

The coding and the code frames

The coding was performed manually, reading through each text response, and assigning it to categories (codes), developing the list of categories at the same time.

The data was only coded once. Errors identified in subsequent analysis were not fixed, as the goal was to replicate the typical quality of coding found in industry, which routinely does contain errors.

Data from the first four weeks of the study (n = 895) was manually categorized, as shown in the table below. On average, each person has been categorized as being in 1.22 categories.

	%	Count
Price	34%	301
Service/Coverage/Network	36%	322
Nothing	14%	122
Everything	8%	72
I like them	5%	45
Customer service	9%	78
Payment arrangements	3%	26
Phone	4%	35
Speed	2%	20
Contracts/No Contracts	2%	15
Other	4%	32
Garbage	3%	24

Likes - Manually coded SUMMARY
sample size = 895

Assessing predictive accuracy

As discussed in the previous section, predictive accuracy is assessed by using cross-validation (i.e., training the predictive algorithm on a sub-sample, and then checking it on a separate test sample). Three metrics are widely used when cross-validating text categorization and all three are reported in the results.

Prediction accuracy

Prediction accuracy is the proportion of responses that are categorized in the correct category. That is, for each of the categories compute the percentage of the test data that was manually coded as being in the category and was predicted to be in the category, plus, the proportion that was not in the category and was predicted to not be in the category.

As the categorizations used in the case study are overlapping, with text responses being permitted to be in multiple categories, the accuracy is computed for each category and then the overall accuracy is the average of these results.

Cohen's Kappa

A practical problem with prediction accuracy being used to evaluate coding is that with smaller categories, a high level of accuracy can be obtained with ease. For example, 4% of people mentioned the phone was a reason for liking their phone company. A model that predicts that 0% of people said "phone" would have a 96% accuracy!

For this reason, the academic literature which focuses on the accuracy of text coding tends not to use accuracy and instead uses a measure known as *Cohen's Kappa,* which takes this problem into account. Scores of greater than 0.8 are usually regarded as being excellent or near perfect.[1]

A natural question to ask is "why not strive for an accuracy of 1?". When two human beings categorize the same data, they will differ in their answers. Partly this is due to the sheer boredom of coding. Partly it is due to subjectivity. And partly due to the ambiguity of many text responses making accurate categorization impossible. As humans cannot achieve perfect categorization it is not sensible to expect a machine to be perfect either. And, an advantage that a machine has over a human is that even though it will make mistakes, it tends to do so in a consistent fashion making comparison over time more reliable.

[1] For a review, see "Inter-rater reliability and coding consistency", Quirk's Marketing Research Review, November 2014 (https://www.quirks.com/articles/inter-rater-reliability-and-coding-consistency).

F1

An alternative metric to Cohen's Kappa is called *F1.* We present the metric as it is widely used in machine learning literature. However, please note that the conclusions are the same regardless of which metric is used, so we focus only on Kappa in our presentation of results.

Results

> The machine learning algorithm is found to be highly accurate at predicting the categorization, suggesting that as few as 200 responses needed to be manually coded to train an accurate model.

Predictive accuracy of the entire categorization

The table below shows the predictive accuracy of the categorization. When the data is trained on 50 respondents, the accuracy of the model is assessed by comparing its predictions on the 845 respondents not used to train the data (i.e., the *Test sample)*. The prediction *Accuracy* is 94.3%. This sounds impressive but is due to the small categories (i.e., as discussed above when you have small categories the prediction accuracy is misleading. Kappa is 0.634, which is well below the 0.8 threshold for near-perfect categorization.

As the size of the estimation sample is increased, Kappa also, on average, increases. Around a sample size of 500 and larger, it crosses the 0.8 threshold.

Training sample size	*Test sample size*	*Accuracy*	*Kappa*	*F1*
50	*845*	*94.3%*	*0.634*	*0.969*
100	*795*	*94.9%*	*0.681*	*0.972*
150	*745*	*95.5%*	*0.728*	*0.975*
200	*695*	*95.6%*	*0.723*	*0.976*
250	*645*	*95.9%*	*0.756*	*0.978*
300	*595*	*95.8%*	*0.753*	*0.977*
350	*545*	*96.3%*	*0.774*	*0.979*
400	*495*	*96.2%*	*0.772*	*0.979*
450	*445*	*96.3%*	*0.778*	*0.980*
500	*395*	*96.0%*	*0.761*	*0.978*
550	*345*	*96.8%*	*0.800*	*0.982*
600	*295*	*96.2%*	*0.774*	*0.979*
650	*245*	*96.9%*	*0.813*	*0.983*
700	*195*	*96.2%*	*0.778*	*0.979*
750	*145*	*96.1%*	*0.769*	*0.979*
800	*95*	*96.5%*	*0.799*	*0.981*
850	*45*	*93.9%*	*0.652*	*0.966*

Accuracy when using only the larger categories

Achieving high predictive accuracy with all the categories is an innately difficult task:

- The category *Other* is difficult to predict. It is the nature of such a category that it contains responses with little in common, and such data is inherently not suited to accurate predictive modeling.
- A category called *Garbage* has been created for people that have provided unhelpful responses (e.g., cursing, irrelevance). That is, these responses have *not* been set as missing values. This is important when using predictive models, as missing data can mean data that is garbage, or, has yet to be coded, so by creating an explicit category for garbage responses, this problem is avoided. As with the *Other* category, this category is inherently difficult to predict.
- Some of the categories are very small. This presents a challenge for predictive algorithms, as small categories mean there is a limited variation which means there is limited information for training the algorithms.
- Many commercial companies have rules regarding the size of categories, such as that categories should contain at least 10% of the sample.

One way to assess the impact of these problems is to exclude *Garbage, Other,* and all the categories with less than 10% of the responses, leaving just the three big categories of *Price, Service/Coverage/Network,* and *Nothing.*

The table below shows the resulting predictive accuracy. This is even more compelling than the previous table, with a sample of around 200 being sufficient for accurate categorization.

Training sample size	*Test sample size*	*Accuracy*	*Kappa*	*F1*
50	*845*	*87.6%*	*0.667*	*0.921*
100	*795*	*90.4%*	*0.753*	*0.935*
150	*745*	*91.2%*	*0.780*	*0.939*
200	*695*	*92.7%*	*0.813*	*0.950*
250	*645*	*92.8%*	*0.814*	*0.951*
300	*595*	*91.3%*	*0.781*	*0.940*
350	*545*	*92.9%*	*0.819*	*0.952*
400	*495*	*93.4%*	*0.833*	*0.955*
450	*445*	*93.0%*	*0.828*	*0.952*
500	*395*	*92.8%*	*0.820*	*0.951*
550	*345*	*93.4%*	*0.832*	*0.955*
600	*295*	*93.3%*	*0.830*	*0.955*
650	*245*	*92.8%*	*0.823*	*0.950*
700	*195*	*95.6%*	*0.887*	*0.970*
750	*145*	*95.2%*	*0.877*	*0.967*
800	*95*	*95.1%*	*0.876*	*0.966*
850	*45*	*96.3%*	*0.904*	*0.968*

Productivity

> The results in the previous chapter demonstrate a productivity gain of at least 134%. In this section, we explain the productivity gain achieved by using machine learning to automate the categorization.

Simple calculations of productivity

A very optimistic calculation of productivity is achieved by focusing on the accuracy with only the larger categories being predicted, which showed that a sample of 200 was sufficient to accurately categorize all 2,090 responses. This suggests a productivity gain of 945% (i.e., 2090 / 200 – 1).

With all the categories being predicted, the productivity gain is 318%.

However, in practice, you need to guess how much of the data to manually categorize before performing the categorization. In conducting this case study, 895 responses were manually categorized, so a more accurate simple calculation is that the productivity gain was 134%.

A more realistic appraisal of productivity

The calculations above are simple, but they are simple in a way that underestimates the productivity gain. In a tracking study, for example, manual coding affects productivity in the following ways:

1. The time it takes to code the data.
2. The tools used to code the data. In particular, if performing traditional manual coding, this is slow, but if using semi-automated coding in Displayr, this time is reduced dramatically.
3. Organizing somebody to do the coding.
4. Exporting and importing the coded data (if using Excel or a standalone program).
5. The time lost while waiting for the earlier previous steps to be completed.

The simple calculations performed above only take the first of these into account. When using machine learning to automate the coding the last three can be entirely skipped, thus increasing productivity further.

Displayr instructions

> This, the final section of the white paper describes how to use machine learning in Displayr to automatically categorize the data.

The process illustrated in this paper is entirely automated in Displayr. The process is:

1. Manually code a subset of the data. This can be done in Displayr (**Anything > Advanced Analysis > Text Analysis > Manual Categorization/Semi-Automatic Categorization**) or in another program and imported into Displayr.
2. **Anything > Advanced Analysis > Text Analysis > Automatic Categorization > Unstructured Text**.
3. Select the variable containing the text data in **Text variable.**
4. Select the variable(s) containing the manual coding in **Existing categorization**. After a few minutes wait the automatic categorization will be completed, and a table will be created at the bottom showing cross-validation results by different sample sizes, as shown below.
5. Save the categorization using **Inputs > SAVE VARIABLE(S) > Categories**.

Automatic Categorization

	Category	Observed (n)	Predicted (n)	Accuracy (n)	Example
1.	Price	40% (301)	41% (698)	100% (301)	▸ "Price, coverage and lack of downtime"
2.	Service/Coverage/Network	43% (322)	43% (728)	100% (322)	▸ "It is reliable"
3.	Nothing	16% (122)	16% (265)	100% (122)	▸ "Nothing"

n = 2090 cases were used in the text processing; existing categorization classified 895 cases into possibly 1 or more of 3 categories; overall training sample performance - Accuracy: 100.0%; Cohen's Kappa: 1.00; F1:1.00; table below shows performance on test samples.

Training sample size	Test sample size	Accuracy	Kappa	F1
50	845	87.6%	0.667	0.921
100	795	90.4%	0.753	0.935
150	745	91.2%	0.780	0.939
200	695	92.7%	0.813	0.950
250	645	92.8%	0.814	0.951
300	595	91.3%	0.781	0.940
350	545	92.9%	0.819	0.952
400	495	93.4%	0.833	0.955
450	445	93.0%	0.828	0.952
500	395	92.8%	0.820	0.951
550	345	93.4%	0.832	0.955
600	295	93.3%	0.830	0.955
650	245	92.8%	0.823	0.950
700	195	95.6%	0.887	0.970
750	145	95.2%	0.877	0.967
800	95	95.1%	0.876	0.966
850	45	96.3%	0.904	0.968

Preventing re-categorization

If performing this task on a tracker, as each wave is updated, this can cause some changes in previous categorizations. In theory, this is a good thing as more information is being used to the accuracy will typically be improving. However, in some contexts this is undesirable. The solution to this problem is to:

1. Set calculation to **Manual** on the automatic categorization output.
2. When each new wave is added duplicate the R Output that contains the automatic categorization, and press **Calculate**
3. Save the variable

4. Create new variables that merge the different variable created in each wave. Please contact support@displayr.com if you need any assistance.

DISPLAYR

Want to cut your analysis and reporting time in half?

See Displayr in action →