Establishing Validity in Content Analyses of Risky Health Behaviors Portrayed in Media Messages

W. James Potter

Department of Communications

University of California at Santa Barbara

Establishing Validity in Content Analyses of Risky Health Behaviors Portrayed in Media Messages

Validity in content analysis is, in essence, the concern over how well coders capture in their coding the meaning that designers intend. But achieving validity is not a simple categorical task; instead it is a complex task that is accomplished by degrees. Validity is built by degrees to the extent that connections are strengthened between key pairs of elements in the coding task. In this essay, I will illuminate what those key pairs of elements are and will show how connections within those key pairs can be strengthened, thereby maximizing validity. To lay the foundation for this discussion, I first lay out three foundational assumptions.

I. Foundational Assumptions

In this section, I will illuminate three distinctions that are foundational to the consideration of validity of content analyses. The first deals with the nature of the content being coded and makes a distinction between manifest and latent content. The second deals with the meaning process and makes a distinction between meaning matching and meaning construction. And the third deals with task specification and makes a distinction between fully and partially specified tasks.

A. Manifest and Latent Content

Manifest content is typically regarded as content elements that are relatively discrete symbols that lie on the surface of the content and are easy for coders to recognize. In contrast, latent content is something – like a pattern or a theme – that must be inferred from the discrete symbols. For example, let's say we want to look at risky behaviors of teens as portrayed on television shows, and we are most concerned with the element of humor in those portrayals. We could treat humor as manifest content and direct our coders to focus on whether there is a laugh track on a television program. When coders hear a laugh track behind a risky portrayal, they code it as humorous; when there is no laugh track, the portrayal is coded as serious. This is a relatively

easy task of recognition and recording of manifest content.

Humor, however, is often a more complex phenomenon that cannot be clearly signaled by a single, salient element in the content. Oftentimes, the experience of humor lies a subtle pattern of elements, that gradually sets up a particular expectation in viewers, then surprises viewers with a clever alternative perspective. This is especially the case with satirical or ironic humor.

When we shift from manifest to latent content, the role of the coder changes from that of a clerk who records occurrences of elements in the content and requires the coder to be more of an interpretive instrument. When the task is to code manifest content, coders need to be provided with some simple prescriptive rules that are based largely on denotative meanings. For example, coders read newspaper articles and make a mark on a coding sheet every time a source is quoted in a story. Also, coders may be asked to select a code from a list of sources by type (e.g., 1 = government official, 2 = spokesperson for a commercial business; 3 = spokesperson for a non-profit organization; 4 = private citizen, etc.) and record the type of source on the coding sheet. Another example is coders watch an hour of television and make a mark on a coding sheet every time there is a commercial ad. Also, coders may be asked to select a code from a list of type of commercial advertiser from a list and record the type of advertiser on the coding sheet. With manifest content, the content elements are fairly salient and discrete. The coder is only asked to recognize certain content elements and record their presence. The skill used by coders is simple deduction, where the coding rule is the major premise, the content elements are the minor premises, and the coder deduces a conclusion of whether the content element fits the rule or not.

When the task is to code latent content, coders must be provided with a very different kind of rule -- one that can guide them in an inductive process of surveying symbols in the flow of content and inferring patterns, which are perceived connections among the individual symbols. Coders are provided with rules to guide them in searching for certain content elements while ignoring others. Coders are also provided with examples that illustrate how particular configurations of elements should be judged as a pattern, but these examples can never be

exhaustive, that is, coders will be continually confronted with novel configurations of elements in the content they must code, and those coders must judge whether the configurations are alike enough to the example in order to be assigned the particular code suggested by the example. Clearly, the coding of latent content requires much more mental effort from coders as they make their judgments. Also, the judgment-making process is susceptible not just to fatigue but it is also sensitive the relative degrees of skill across coders (ability to perceive content elements and ability to construct patterns systematically) as well as coders' idiosyncratic knowledge about the content being coded. The threats to reliability and validity are much more severe with coding latent content compared to manifest content.

Despite the challenges inherent in coding latent content, there are big payoffs if it is done well. For example, it is more valuable to know how much dangerous driving behavior there is in television shows than to know how many times characters fastened their seat belts. It is more valuable to know how much sexual activity there is in media stories than to know how many times one character touched another character.

B. Meaning Process

There is an important difference between meaning matching and meaning construction. Meaning matching is a relatively automatic process where people recognize content elements and quickly locate the denoted meaning in their memory. In contrast, meaning construction is a much more involved process where people need to exercise judgment as they infer patterns across many elements.

This distinction is related to the previous one about manifest and latent content. What makes the coding of manifest content typically so efficient and reliable is that it is an automatic process. The coding of latent content is much more complex because coders must construct meaning rather than simply match it to their previously learned meanings. To the extend that all coders follow the same procedures in constructing meaning, the outcomes of those meaning construction processes will be similar and hence the reliability of the coding with be high. Also, this distinction has relevance for validity. With meaning matching, coders make connections

between content elements in the media content and previously learned denoted meanings. Because it is likely that all coders share the denoted meanings, the potential for validity is very high. However, with the meaning construction process, coders are susceptible to several factors that can lead to divergence in the product of the meaning construction process. Major among these factors are differences in skills of perceiving content elements and in inferring patterns across those content elements); degree of drive for completeness and accuracy; and understanding of as well as willingness to follow all the coding rules consistently.

In short, with the task of coding manifest content, coders are "decision-makers," but with latent content, coders are "judgment-makers." Decision-making is the selection among a finite set of options in a relatively automatic, standard manner. Coders rely on simple prescriptive rules that are based largely on denotative meanings. The coder is only ask to recognize certain content elements and record their presence. The skill used by coders is simple deduction, where the coding rule is the major premise, the content elements in the content are the minor premises, and the coder deduces a simple conclusion of whether the content element fits the rule. In contrast, judgment-making requires the use of an inductive process of surveying content elements in the flow of content and inferring patterns, which are perceived connections among the individual content elements. Coders are provided with rules to guide them in searching for certain content elements while ignoring other content elements. Coders are also provided with examples that illustrate how particular configurations of content elements should be judged as a pattern, but these examples can never be exhaustive, that is, coders will be continually confronted with novel configurations of content elements in the content they must code, and those coders must judge whether the configurations are alike enough to the example in order to be assigned the particular code suggested by the example. Clearly, the coding of latent content requires much more mental effort from coders as they make their judgments. Also, the judgment-making process is susceptible not just to fatigue but it is also sensitive the relative degrees of skill across coders (ability to perceive content elements and ability to construct patterns systematically) as well as coders' idiosyncratic knowledge about the content being coded. The threats to reliability and

validity are much more severe with coding latent content compared to manifest content.

C. Task Specification

There is an important difference between fully specified and partially specified tasks. A fully specified task is one where all the information that is needed to solve a problem is available to the person. For example, consider the following problem: 5 + 7 = _____ . This task is fully specified, because a person has enough information to solve the problem. Now consider this problem:

Y + Z = 24. This problem has two unknowns – Y and Z – so there is not enough information to arrive at one solution with confidence. You could answer 6 and 18 while I might answer 10 and 14, and we would both be right. There are also many other correct answers to this problem, because it is only partially specified. To most of us, each of these "answers" intuitively seems faulty. Because many answers are possible, can any one of them be regarded as "the solution?" Key information is missing from the problem that once included would help us all arrive at the one and only one correct solution with confidence.

It is helpful to think of the coding of manifest content as a fully specified task that requires meaning matching. This task is fully specified when content analysis designers construct simple rules to direct coders to certain content elements and believe that the application of these rules sets up a clean meaning matching process. Most social scientists designing content analysis attempt to craft their code books to make the task fully specified; they believe that reliability and hence validity will be relatively low to the degree that the task is left as partially specified. This, of course, makes sense but *only if the content being coded is manifest*. I argue that the goal of fully specifying a coding task is not possible with latent content and furthermore, steps taken to try to fully specify the coding of latent content will have the effect of reducing the validity of the generated data.

Does this mean that the coding of latent content is hopeless or that it can never be achieved in a scientific manner? My answer is no. The coding of latent content is important, and I would go as far as to say that the coding of latent content is far more important than the coding

of manifest content. However, when social scientists focus on latent content, they need to approach the construction of the code book and the training of coders is a very different way than when they focus on manifest content. In the next section I will suggest a strategy to guide content analysis designers through the complex challenges of achieving high validity in their coding.

II. Strengthening Connections

Recall from the introduction that I argued that validity is built by degrees to the extent that connections are strengthened between key pairs of elements in the coding task. In this section, I will focus on three key pairs of elements. First, there is the connection between the coding rules and the content being coded. Second, there is the connection between the coding rules and coders' existing knowledge. And third, there is the connection between scholarly meaning and everyday meaning. Validity is the degree to which these connections are strong. I suggest that content analysis designers need to follow a strategy that maximizes this set of three connections.

A. Connection between Coding Rules and Content

The connection between the coding rules and the content being coded is typically the focus of all social scientific content analyses. Designers spend a great deal of effort in crafting a code book that is able to direct coders to look for the right things in the content. With manifest content, this task is usually enough because a good coding rule provides a sufficient condition for recognizing a content element and matching it to its proper code.

With latent content however, the code book needs to provide guidance in two areas: recognizing elements in the content and assembling those elements into some sort of pattern in order to make the coding judgment. Each coding judgment usually requires the recognition of more than one content element so the code book needs articulate all the content elements that need to be recognized, then it also needs to guide coders into doing something with those elements to arrive at a coding judgment. Thus the designing the code book for latent content is significantly more challenging than designing a code book for manifest content. The challenge to the recognizing elements task lies in conceptualizing the full range of elements that must be

attended to; this is typically an impossible task to fully specify. Therefore these recognition rules must be suggestive instead of definitive. But the even greater challenge is in providing the calculus in a way to guide all coders in the construction of a coding decisions, *even when all coders do not have the same recognition set*. In order to do this, I recommend code book designers begin by thinking about what content elements are necessary and which are substitutable. A necessary element is that which must be present in the content in order to qualify for a particular code. An example of this type of rule is: Code the content if it includes the elements of A, B, and C. In this coding rule, there are three necessary elements. This is a very different rule than: Code the content if it includes either the element A, the element B, or the element C. This later rule has three substitutable elements; the presence of any one of these three elements qualifies the content for coding.

In the previous paragraph, the elements were also presented as positive conditions, that is, their presence in the content leads to the need to code the content. Conditions can also be written as negative, that is, the presence of a negative condition in the content leads to it not being coded. A negative condition coding rule would be: Do not code content if A appears in that content. Negative conditions can present a series of necessary elements (i.e., do not code content if A, B, and C appear in that content) or substitutable elements (i.e., do not code content if A, B, C appear in that content).

It is possible to use positive and negative conditions as well as necessary and substitutable elements in a single coding rule (i.e., Code the content if A and B are both present along with either E, F, or G; but do not code the content if either X or Y or present along with either Z or W). It is easy for designers of coding schemes to get carried away and create some very complex coding rules. There is a natural desire to go in this direction by social scientists who want to provide their many coders with enough guidance so that they all make the same decisions when presented with the same coding tasks. However, it is best to keep in mind that there is a curvilinear relationship with number of rules and validity. The more rules, the greater the danger of losing ecological validity (see the third connection in this section) and of

complicating the coding task to a point where coders lose focus on what is most important. I suggest that designers try to work to a maximum point on the validity plot (curvilinear relationship), then shift attention away from the connection between coding rules and content and to the connection between coding rules and coders' existing knowledge, which bring us to the next topic. The designer of coding rules needs to be careful not to "over-rule" the task. There is a value to writing enough rules to provide clarity in guidance but not too many rules that would confuse coders. Designers must ask if the addition of a guidance rule has marginal utility, that is, if the addition of another rule and its subsequent addition in complexity will be paid back in terms of increases in coding agreement. If the answer is yes, the guidance rule has marginal utility and should be included. If the answer is no, then the increase in complexity would result in less, not greater coder agreement, and hence should not be added. There are projects that are so driven in their sincere quest to provide coders with enough guidance so they can be consistent in their coding that they end up being "over-ruled." By this I mean that beyond a certain point, the addition of coding rules drives the costs of the study beyond its benefits.

B. Connection between Coding Rules and Coders' Existing Knowledge

The connection between the coding rules and coders' existing knowledge is typically overlooked in designing code books, however, all designers make implicit assumptions about this connection. For example, with the coding of manifest content, code book designers assume coders will be able to recognize content elements and all attach the same denoted meaning to those symbols. To illustrate this point, recall the example above about a fully specified problem of: 5 + 7 = ___ . The designers of this problem did their part to fully specify the problem by leaving only one unknown. However, we should not ignore what the solvers of the problem must bring to the task. Solvers of this problem must bring to it their mathematical schema in order to understand that "5" and "7" represent numbers with particular values and that the content element "+" means addition. Even though the designers of the problem constructed it with only one unknown, the problem is not fully specified *unless people come to it with a good mathematical schema.* If people do not have this schema, the designers of the problem need to

provide additional rules in order to fully specify the task, that is, they must train people in concept of number addition.

With the manifest content coding task, we assume our coders bring to the task a standard set of denoted meanings for the content elements we are asking them to code. With the latent content coding task, we should make the same assumption. However, the difference is the perception that what coders bring to the task is not shared among them to a sufficiently high enough degree for designers of the code book to trust that coders will all make their decisions the same. But this perception may be wrong, that is, there may be many types of latent content for which coders all construct the same meaning. This is likely to be the case with media messages, because those messages are usually highly formulaic, and most people in our culture have been exposed to these formulas so often that the processing of meaning of these messages has become highly routinized.

With content analysis projects, we must ask ourselves: What schemas do people bring to the coding task? Rather than regard this schema information as idiosyncratic and a source of potential error in coding, we need to examine coders for the consistency in shared schemas and hence natural uniformity in meaning construction. Thus what coders bring to the task can be regarded often as a powerful source of validity rather than a threatening source error. The challenge for code rule designers is then not to re-channel coders into a special way of constructing meaning, but instead to cue coders to access their natural ways of constructing meaning.

If we do not consider this, we are at minimum making our design task much more difficult than it need be and at maximum we are writing large rule sets to try to overcome people's naturally occurring schema and thereby destroying claims for ecological validity of our results. When social scientists design a content analysis of latent content, they need to consider carefully what coders bring to the task.

C. Connection between Scholarly Meaning and Everyday Meaning

Finally, we come to the connection between scholarly meaning and everyday meaning,

which is the concern about ecological validity. If we spend too much effort working on the connection between rules and content while spending too little effort on the connection between rules and what coders already know, we are likely to create a problem with ecological validity.

Problems with ecological validity are especially troublesome with media messages, because most people have had so much experience with media messages, they have developed perceptions of the content that are strongly held and reinforced over many years. When social scientists use scholarly definitions that are different from the public's everyday definitions for content, the results of those scientific content analyses are going to look very strange to the public. An example of this is with the coding of violence on television. Scholars focus on the occurrence of aggressive acts that harm or have the potential to physically harm other characters. However, the public also considers context and includes the negative conditions of humor and fantasy such that when a violent act occurs in a cartoon, it is not violent because of the humor and fantasy. So when scholars report finding high rates of violence on television, they are referring to something quite different than what the public regards as violence. This is most apparent when the public is told that cartoons are by far the most violent type of programming on television.

I am not arguing that the public's everyday meaning is superior to that of scholars and that scholars should abandon their meanings. Instead, I am pointing out that often the two groups have different meanings to serve different purposes. Each set of meanings can be good for its own purposes. The problem arises when we take the meanings from one group and apply it in the other group without clarifying the differences in definition and purpose. When scholars report their results to the public without carefully articulating their meanings and presenting those meanings in a way that resonate with the public, those results have little ecological validity.

Conclusion

The fundamental purpose of crafting coding rules is to achieve full specificity of the coding task. If the content analysis is to generate scientific data, the research study must be designed so that all coders are given enough direction so that coding decisions are made

consistently across coders, across content, and across time. Consistency is the key to reliability, and reliability is one of the major components in achieving validity in the data. The key to achieving consistency lies in fully specifying the coding tasks.

There is more than one way to fully specify the coding task. One way is to strengthen the connection between coding rules and the content. This is the typical technique used by social scientists and it shows up in the addition of more coding rules and a higher degree of complexity in those rules.

When we approach content analysis from a social science perspective, we do not want each coder to arrive at his/her own unique interpretation of the content, even if each coder's judgment is a reasonable and defensible one. Instead we want convergence; we need to provide enough rules so that all coders make the same decisions and all arrive at the same "solution." When we pilot test our initial design and find that the consistency in judgments across coders is low, our natural response is to assume the task is not yet fully specified and we need to construct more rules. However when we do construct more rules and re-train the coders, the consistency might not increase; sometimes it decreases. There are several reasons for the possible decrease. One reason is that the additional complexity puts too much of a burden on coders, so fatigue increases. Another reason is that a large rule set might serve to *over*-specify the task, that is, the greater number of rules raises an even greater number of sub-questions that are left un-answered. Yet a third reason is that coders struggle with the dialectic between the rule set and their own intuitive understanding of the phenomenon they are asked to code.

I argue that we as social scientists can approach full specification on most content analysis projects. However, the strategy to close the specification gap rests less with the number of rules and more with the type of rules we construct. There is a trap in the design of code books for designers to continually increase the number -- and complexity -- of content coding rules. However, there is a point where designers need to shift to a focus on the connection between the coding rules and coders' existing knowledge. A focus in this area can make up for some of the deficiencies in the previous one where the focus is on the content itself. Finally, designers need

to work on strengthening the connection between scholarly meanings and everyday meanings. Only by working on strengthening all three types of connections will designers of content analyses have the potential to maximize the degree of validity of their findings.