



Using Topic Modeling for Code Discovery in Large Scale Text Data

Zhiqiang Cai^(✉) , Amanda Siebert-Evenstone , Brendan Eagan ,
and David Williamson Shaffer 

University of Wisconsin-Madison, Madison, WI 53706, USA
zhiqiang.cai@wisc.edu

Abstract. When text datasets are very large, manually coding line by line becomes impractical. As a result, researchers sometimes try to use machine learning algorithms to automatically code text data. One of the most popular algorithms is topic modeling. For a given text dataset, a topic model provides probability distributions of words for a set of “topics” in the data, which researchers then use to interpret meaning of the topics. A topic model also gives each document in the dataset a score for each topic, which can be used as a non-binary coding for what proportion of a topic is in the document. Unfortunately, it is often difficult to interpret what the topics mean in a defensible way, or to validate document topic proportion scores as meaningful codes. In this study, we examine how keywords from codes developed by human experts were distributed in topics generated from topic modeling. The results show that (1) top keywords of a single topic often contain words from multiple human-generated codes; and conversely, (2) words from human-generated codes appear as high-probability keywords in multiple topic. These results explain why directly using topics from topic models as codes is problematic. However, they also imply that topic modeling makes it possible for researchers to discover codes from short word lists.

Keywords: Coding · Grounded coding · A priori coding · Automatic coding · Grounded theory · Qualitative analysis · Quantitative analysis · Latent semantic analysis · Topic modeling · Code discovery

1 Introduction

One of the most important steps in a quantitative ethnographic analysis is linking evidence to meaning. In general, coding is an analytic process that searches for relevant features within a dataset and assigns meaning to a given piece of evidence [8, 29]. There are both many traditions of coding as well as many ways that researchers engage in the process of coding. Researchers may generate codes from data, theory, or a literature review of a field. In this process, researchers engage in a number of steps and make many decisions. For example, in a single analysis, researchers make decisions about the underlying data including choosing, cleaning, segmenting, assigning meta data, and other choices that affect the resulting data and codes. Further, researchers perform many

operations on a code or a set of codes including identifying, naming, organizing, categorizing, aggregating or disaggregating, refining, validating, and applying the codes. Throughout this process, the ultimate goal of coding is to engage in a rigorous procedure to warrant a meaningful story about the data [33].

The process of developing codes thus depends on reading data deeply; this is what allows researchers to discover what is happening in a given context. However, this process is quite difficult and time-consuming [24]. Machine algorithms, in contrast, can read data quickly and discover patterns that researchers may have missed. However, machine algorithms read data without understanding its meaning deeply, and thus may produce codes that don't describe the data in useful ways [32].

Fundamentally, we argue that because coding is a process of searching for *meaningful* patterns in data, it may be best suited by combining the talents of humans and machines. In this paper, we explore the iterative process of identifying and developing codes. Then we explore how machine learning methods can supplement code discovery and refinement rather than replace meaning-based processes of coding.

2 Background

2.1 What are Meaningful Patterns?

Building on the idea of *thick description*, or explanations of action in terms of their socially-situated meanings, Geertz (1973) argues that, fundamentally, data analysis “is sorting out the structures of signification” and then “determining their social ground and import” (p. 9) [17]. Consequently, researchers sort and interpret their observations to discover what is happening in a given context, in light of how participants would understand and describe it. As Ryle and Geertz [17, 31] argue, one common way to engage in this process is by creating codes.

Many researchers may employ a *grounded approach* to coding: intensively reading and rereading qualitative data to discover and construct theory and codes from it [18, 32]. Researchers engage in a rigorous and systematic search for meaning [25] to find *patterns*, where some phenomenon repeats in a predictable way. Only by finding and comparing such patterns can a researcher begin to identify and distinguish codes.

Many factors influence how a researcher makes sense of data. Each researcher brings a different set of experiences and identities to their work, which may help them find certain descriptions and miss or disregard others. To address this problem, a common heuristic for developing codes is to start by describing each line of data, which, as Charmaz argues, “helps you to refrain from inputting your motives, fears or unresolved personal issues to your respondents and to your collected data” (p. 37) [10].

A grounded coding process iterates through open, axial, and selective coding. *Open coding*, or coding line by line, breaks down data to identify preliminary concepts. These early descriptions provide the basis for future interpretations [33]. Next, researchers begin making connections between ideas while comparing concepts and categories during *axial coding*. In this step, there is more emphasis on identifying features and conditions that may explain what is happening or provide contextual information. Interpretations become clearer during *selective coding* where researchers build a conceptual framework to understand the relationships among the codes.

Successful coding, then depends of researchers' ability to construct interpretations and build meaningful models. But that ability, in turn, depends on the ability to find systematic patterns in data, and in this aspect of coding, computers are often much more efficient than human coders. A machine can quickly go through a large volume of data to identify patterns or group similar patterns together.

At the same time, the machine learning algorithms often used for such pattern finding suffer from different problems. Chen and colleagues [11] outlined various challenges in using machine learning for qualitative data analysis: such methods are often unable to build useable models, use decontextualized features, require large amounts of data and in some cases pre-labeled data, and tend to perform better on high frequency codes in comparison with more sparse codes. Thus, a machine learning approach to pattern-finding may not result in good codes or interpretations of data: patterns may be systematic, but they may not be meaningful.

In this paper, we argue that good storytelling, and thus good coding, requires identifying meaningful patterns in your data. By combining the systematic pattern-finding abilities of computers with the human ability for meaning-making, we explore how researchers can combine qualitative data analysis with machine-learning to discover codes and refine our coding process.

2.2 Computer-Facilitated Coding

Large scale text-based data is increasingly easier to find. In the field of education, large scale text data is often collected from online learning environments, such as MOOCs [14, 22, 28, 36–38], intelligent tutoring systems [1, 6, 9, 19], virtual internship systems [12, 13, 35], and others.

Broadly speaking, two classes of computational tools have been developed to help researchers code large scale data. One class of tools copy manual, paper-and-pencil-based processes into a screen-and-mouse environment. Manual coding systems such as NVIVO and MAXQDA support coding through better data organization, quicker data navigation, more convenient ways of highlighting data segments, and visual displays of the coding process and results. While it is more convenient to use such tools than paper-and-pencil, researchers still have to go through the data line-by-line to mark all possible instances of a code.¹ Coding systems like the Reproducible Open Coding Kit [27] help users prepare and annotate data for future modeling, including data that was manually coded using one of the systems listed above.

A second class of computational approaches includes tools such as the nCoder [8, 15] and LightSide [23], which augment the coding process by introducing statistical and machine-learning tools. In the next section we explore how this class of tools affects the coding process by describing the functionality of nCoder.

2.3 Coding and Validating Through nCoder

nCoder is a tool that helps researchers develop, validate, and apply automated classifiers to their data. It uses an active machine learning approach aimed at maximizing the impact

¹ Some of these systems also include rudimentary keyword-based searches to support coding.

of human coding expertise while minimizing the researchers' effort. nCoder has been used in analyses in an array of fields including studies of engineering [2], naval warfare teams [34] as well as on a variety of data types, such as chat logs [12], interviews [21], and training session transcripts [30]. nCoder augments the coding process in 4 important ways.

1. *nCoder scaffolds the generation of classifiers.* As in traditional qualitative analyses, a researcher names and defines conceptual ideas or patterns. Users then create a procedural (algorithmic) definition of a given code using *regular expressions* (regexes): words, word stems, or more complex patterns of words. For example, a user interested in how students define ideas may search for “definition”, look for all words that contain the stem “defin”, or explicitly search for words that start with “defin” but do not include the common colloquialism “definitely”: `^(?!definitely)*\bdefin`. Next, nCoder provides data samples taken from a training set for the rater to use as a guide when they apply and revise their regexes.
2. *nCoder modifies the coding process by changing the way data is sampled.* Specifically, nCoder draws a test set of size n from the holdout set using a conditional sampling method called *base rate inflation*. nCoder randomly selects items one at a time and codes them. Selection and coding continue until $20\% \times n$ of the items match one of the regexes for the code. These positively coded items are added to the test set. Then nCoder randomly selects an additional $80\% \times n$ items, codes them, and adds them to the test set. This ensures that even with small test sets and/or low frequency codes, raters will see instances that represent the concept being coded. Once the human rater has coded the test set and inter-rater reliability has been computed, the test set items are added to a training set for users to refine the regexes.
3. *nCoder provides a three-way validation process to ensure that codes are both procedurally and conceptually valid.* This coding process involves three raters – two human raters and a machine rating codes using regex matching. One human rater starts the coding process by defining codes. For each code, the rater creates a classifier for the code by specifying a set of regexes. The human rater then codes a test set to see whether or not the examples of the code that were found using the regexes match their understanding of that concept. Cohen's kappa is calculated between the human rater and the machine establishing reliability or procedural validity. After achieving validity for this procedural definition of the code, the rater measures their agreement about the code with a second rater, establishing conceptual validity. Only when all three raters are in agreement by achieving an acceptable kappa and rho (described next), the classifier is validated.
4. *nCoder calculates Shaffer's ρ to warrant interrater reliability of codes on a large dataset from small test sets.* Shaffer's ρ is a Monte Carlo rejective method that tests whether an interrater reliability statistic is over a given threshold at a specified α level [15, 32]. That is, Shaffer's ρ tests whether it is possible to claim that the population κ is above some threshold τ based on the κ value in a test set.

These features allow a researcher to use nCoder to manually code a small amount of data and develop reliable and valid classifiers to automatically code large scale data.

2.4 Using Topic Modeling for Code Development

While existing tools help researchers *code* their data, they may lack the power of discovering *potential codes* in large scale data. Advances in *machine learning* and *natural language processing* (NLP) make it possible to automatically discover patterns that humans cannot see in data, or specific instances of existing codes that humans might miss. Topic modeling [5, 7] is one of the most popular NLP methods for automated coding. Topic modeling uses a large corpus of data to generate groups of words, called *topics*, each of which is represented by a probability distribution assigned to the words in the data. The words with highest probabilities are used to interpret the “meaning” of the found topics, where we put meaning in quotation marks to indicate that this is a post-hoc description, assigned by a researcher, to one of the groups of words identified by the topic model. Topic modeling also assigns *topic proportion scores* to each document which are often used as non-binary codes.

The problem is that topics from topic modeling are not always easy to interpret and topic model-based coding is hard to validate. There has been research comparing human coding with topic model based coding [3, 4, 26]. But while the goal of such studies is to improve topic interpretability, no study has been able to match human coding well enough that human coding can be replaced. Topic modeling was originally designed to be a *generative method* to help researchers find and describe what is in large numbers of text documents [5]. That is, it was intended to help surface underlying and undiscovered patterns in data. However, as Bakharia [3] and others argue topic modeling only produces “buckets of words” that may not help researchers tell meaningful stories.

However, the fact that topic models cannot replace human coders does not mean the method has no role to play in developing meaningful codes. Researchers have identified several ways that topic modeling could a “valuable aids within the quantitative ethnography process” (p. 297) [3]. In this study, we use topic modeling as a discovery tool for researchers to identify and refine codes and then develop reliable and validated classifiers to code data. Our work is closely related to the nCoder coding process. We ask:

RQ1: Are human created codewords (words that match the regexes) grouped by topics? This question asks whether or not a set of code words for a given code appear together at the top (sorted by probability from high to low) of a single topic.

RQ2: Do high-probability keywords in topics include codewords identified by human coders? This question asks whether or not human created codewords emerge at the top of the topics (not necessarily grouped together).

To answer these questions, we used two existing datasets for which researchers have established and validated regex-based coding classifiers through nCoder. We then ran topic models on the same datasets and find human created codewords that appear as high-probability words within the topics identified.

3 Methods

3.1 Data

Our study examined two existing datasets. The first dataset was from a study by Ruis and colleagues (2019), which we will refer to as the *medical dataset* [30]. Their study investigated the use of procedural simulation in continuing medical education short course. The study included 58 surgeons who participated in a one-day course on laparoscopic inguinal and ventral hernia repair. Participants were assigned to groups based on their reported experiences. Their group discussions were recorded and transcribed. The transcribed data was coded with six codes (see Table 1), which were defined based on ethnographic observations and conventional content analysis. Regex-based classifiers were created and validated through the nCoder tool. Table 1 shows the definition of the codes, the number of codewords involved, and the validation results.

Table 1. Definition and validation of codes in the medical dataset

Code	Description	Codewords (74)	Human 1 vs Human 2	Human 1 vs Computer	Human 2 vs Computer
			κ^*	κ^*	κ^*
REAL WORLD CASE	Referencing real bodies, patients, or other cases	8	0.80	0.95	0.73
MESH REPAIR	Referencing mesh, tacking, or suturing	27	1.00	1.00	0.97
GENERAL ANATOMY	Referencing the anatomy of the abdomen	12	1.00	1.00	0.96
PATHOLOGICAL ANATOMYA	Referencing the anatomy of a hernia	27	0.95	0.98	1.00
REQUESTING ADVICE	Asking what surgeons should do in a given situation	–	0.86	0.86	0.75
TROUBLE- SHOOTING	Managing or negotiating complications	–	0.85	0.89	0.80

* All kappas are statistically significant for $\rho(0.65) < 0.05$.

The second dataset was collected from the engineering virtual internship Nephrotex which we will refer to as the *engineering dataset* [12, 13, 16]. The data was collected

Table 2. Definition and validation of codes in the engineering dataset

Code	Description	Codewords (232)	Human 1 vs Human 2	Human 1 vs Computer	Human 2 vs Computer
			κ^*	κ^*	κ^*
TECH CONSTRAINTS	Referring to inputs: material, processing method, surfactant, and CNT %	33	0.96	1.00	0.96
PERFORMANCE PARAMETERS	Referring to attributes: flus, blood cell reactivity, marketability, cost, or reliability	87	0.88	0.93	0.84
COLLABORATION	Facilitating a joint meeting or the production of team design products	13	0.76	0.87	0.76
REDESIGN REASONING	Referring to design and development prioritization, tradeoffs, and design decisions	33	0.89	0.86	0.84
DATA	Referring to or justifying decisions based on numerical values, results tables, graphs, research papers, or relative quantities	34	0.94	0.9	0.89
REQUESTS	Referring to or justifying decisions based on internal consultant's requests or patient's health or comfort	32	0.88	0.94	0.94

* All kappas are statistically significant for $\rho(0.65) < 0.05$.

from novice engineering design teams participating in an educational simulation. Previous analyses of the engineering dataset used nCoder to develop and validate six codes and regex-based classifiers. Table 2 shows the definition of the six codes, number of codewords, and the validation results.

3.2 Extracting Codewords

Codewords for both sets were extracted by matching the words in the dataset with the regex-based classifiers. For the medical dataset, the codes “Requesting Advice” and “Trouble-shooting” were removed because they used multiple word classifiers. From the remaining four codes, a total of 74 codewords were found in the data (see Table 1). For the engineering dataset, all six codes were used in this study. The code “data” matched many numbers which were excluded from the codeword list. A total of 232 codewords were found (see Table 2).

3.3 Generating Topic Models

We used the LDA function in the R package *topicmodels* [20] to create topic models. The models depend on several parameters which has impact to the performance of the models. However, we were interested in seeing how well topic modeling works with non-optimal choices, as these are the parameters most likely to be chosen by expert coders who are not expert topic modelers. Thus, instead of optimizing performance, we chose to construct a *naïve topic model*, meaning a model with choices that are reasonable and easy to implement.

Document Segmentation. The average size of the documents and the total number of documents are two competing parameters in tension with one another. On one hand, larger document size results in more accurate probability estimation within documents. On the other hand, larger numbers of documents result in better probability estimation across documents. Data lines can thus be split or merged to get the optimal balance of average document size and number of documents. We chose to use the lines of data without alteration as documents input to the topic model.

Word Filtering and Word Lemmatizing. Word exclusion and lemmatizing are two standard practices in topic modeling. *Word exclusion* leaves out commonly occurring words such as conjunctions, articles, and prepositions. We used word exclusion by removing standard function words (of, the, in, etc.). *Lemmatizing* converts words into their root or *lemma form*: for example, *Chairs* becomes *chair*, *seeing* becomes *see*, etc. However, we used the original words rather than their lemmatized form because lemmatizing is time intensive, particularly when the computational power is limited and data size is large.

Number of Topics. Researchers often struggle to determine an appropriate number of topics when trying to optimize a topic model. We chose 10 topics for both models for practical considerations. Ten is larger than the number of codes (4 for medical and 6 for engineering) we were investigating, we anticipated that some subset of topics could represent the different codes. Ten topics was also small enough for us to easily review for this study.

Based on these choices, topic models were generated for the medical and engineering datasets. For each topic in each model, a word list was generated by sorting the words in the topic by word probabilities and choosing the top 15.

4 Results

Table 3 shows the top 15 keyword lists for the topics in the medical dataset. The highlighted words are codewords developed with human input using nCoder. Four different colors represent the four different codes. Similarly, Table 4 shows top 15 keyword lists for topics from the engineering dataset, with six different colors representing six different codes.

Table 3. Codewords in top 15 topic words for the medical dataset

n	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9	T 10
1	yeah	ll	inaudible	mesh	pull	camera	ah	defect	ve	left
2	don	suture	alright	tack	mesh	balloon	don	mesh	hernia	hand
3	bit	cut	sac	sutures	hard	space	uh	close	patient	feel
4	move	grab	mhm	don	guys	port	peritoneum	midline	inguinal	model
5	yup	hold	push	positioning	ten	lateral	um	time	costal	cooper
6	pretty	needle	laughs	tacks	roll	start	lot	size	margin	dissect
7	angle	measure	didn	position	twenty	trocars	real	hole	repair	scope
8	knife	wall	supposed	tacking	nice	incision	stuff	people	ventral	yep
9	finger	abdominal	easier	system	centimeters	oblique	vas	top	lap	fat
10	lower	bring	white	correct	fifteen	rectus	vessels	tie	patients	scissors
11	trouble	spinal	guy	flat	meshes	perfect	easy	bigger	bowel	thirty
12	table	mesh	cord	fixation	pulling	medial	direct	leave	haven	tacker
13	glove	coming	ring	people	middle	trocars	epigastric	grasper	omentum	ahead
14	knot	mark	demonstrates	stay	makes	external	tapp	piece	lot	pubis
15	spot	inside	guess	bit	centimeter	internal	fascia	tension	laparoscopically	won

Color codes: real word case | mesh repair | general anatomy | pathological anatomy

Table 4. Codewords in the top 15 topic words for engineering dataset.

n	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9	T 10
1	consultants	flux	batch	steric	surfactant	list	guys	notebook	prototype	shared
2	prototype	reliability	team	cnt	steric	notebook	im	task	1	space
3	prototypes	cost	design	phase	agree	internship	yeah	submit	cost	write
4	material	marketability	attach	vapor	hindering	entry	process	alex	2	information
5	internal	bcr	submit	process	surfactants	alex	report	email	prototypes	posted
6	meet	reactivity	testing	prototype	choice	questions	padma	notebooks	3	notebook
7	test	blood	notebook	surfactant	negative	check	time	due	design	consultants
8	agree	cell	team's	dry	graph	deliverable	manufacturing	complete	met	internship
9	design	low	prototype	hindrance	performed	submitted	alan	5	4	alex
10	results	rate	rest	20	charge	ne-phrotox	guess	team	5	time
11	decide	11	option	jet	attributes	deliverables	rudy	submitted	pmma	meeting
12	cosultant	lower	specifications	material	cost	summary	supposed	send	agree	michelle
13	attributes	lowest	results	2	re-search	submit	meeting	time	choices	analysis
14	requirements	43.33	time	pespvp	meeting	clicking	5	forget	standards	engineering
15	5	pmma	prototypes	10	hydrophilic	email	hey	deliverable	attributes	similar

Color codes: tech constraints | performance | collaboration | design decisions | data | requests

4.1 RQ1: Are Codewords Grouped by Topics?

In an ideal topic grouping each topic keyword list should contain either no codewords or only codewords from the same code. In practice, if a list contains a large proportion of codewords from a single code, the topic could represent that code well.

For the medical dataset (see Table 3) the topic T_4 contains the largest proportion of codewords: 40% of the 15 most probable words for the topic are from the code MESH REPAIR. Other topics contain smaller proportions of codewords from any single code.

In the engineering dataset the performance of the topic model on this criterion is even worse. No topic has more than 27% of the 15 most probable words from a single code.

Thus, in both cases, topic keyword lists only contain a small proportion of codewords from a single code; most of the most probable topic keywords are either not codewords for any code, or codewords from multiple codes. This suggests that naïve topic modeling does not do a good job of identifying clusters of codewords.

4.2 RQ2: Do High-Probability Topic Keywords Include Human Identified Codewords?

From Table 3 and Table 4, we see that for both datasets, every code has at least 2 codewords included in the most probable words across all topics. This suggests that

these naïve topic models identify some codewords from all of the human-identified codes.

5 Discussions

The goal of this study is to investigate whether or not topic modeling could help in discovering codes. That is, assuming that a researcher does not already have a set of codes in mind, could a naïve topic model help to discover useful codes to describe the data?

The common practice of using topic modeling is to “label” the topics by reviewing the top keywords. Our study shows that, topic keywords do not group together words that from classifiers that humans developed using traditional approaches to code identification. Thus “labelled” topics from topic modeling are unlikely to replicate codes that a human would identify as meaningful. Using topics as codes, then, may result in misleading conclusions.

However, our results did show a different potential use for topic modeling in code identification. Although the topic modeling keyword lists cannot be used as codes, they may provide keywords that could be used for code discovery.

In our study, all human identified codes had far more codewords included in the keyword lists than would be likely due to chance alone. For example, the code `REAL WORLD CASE` had 8 codewords in the medical dataset. The medical dataset had 4581, unique words, so the likelihood of choosing a codeword that indicates a `REAL WORLD CASE` is $8/4581 = 0.17\%$ or 1 in 573. We examined 10 lists of 15 keywords for the medical dataset. At chance we would expect to find $150 \times 0.17\% = 0.26$ codewords that indicate a `REAL WORLD CASE`. In fact, we found two such codewords—that is, codewords for a `REAL WORLD CASE` were 8 times more likely than at random. Moreover, the 150 high-probability keywords from the topic model identified 27 codewords from the human-identified codes—that is, 18% of the keywords were meaningful for codes in the dataset. This, in turn, suggests that the high-probability keywords from a topic model provide a good source of words that a human coder should consider investigating as possible keywords for codes.

Whether or not a code can be actually found through these codewords is still a problem. It depends on how sensitive a researcher is to the codewords and how strong the codewords signal a code to the researcher examining them. The topic keyword lists could be more useful if the actual data containing the keywords could be easily reviewed. A *topic modeling utility* could create a naïve topic model and let a researcher click on a keyword to see sample of data lines that contain it. The data lines may provide much richer information about whether a particular keyword is a clue to a meaningful code.

This work has the obvious limitation of only investigating two datasets with validated regex classifiers. Also, we deliberately chose naïve topic models. It is possible that other algorithms could be developed to improve topic model performance by automatically choosing more effective parameters for a non-technical user, or using more sophisticated supervised topic modeling. Finally, the work here is based on *post hoc* analysis of codewords as they appear in topic keywords. Future work needs to examine the conversion rate of topic keywords to new codes.

These limitations notwithstanding, this work confirms previous findings that topic modeling is not a good substitute for human coding; however, it also suggests that topic modeling can potentially supplement manual and automated coding methods by helping researchers discover potential keywords for new codes or to augment existing codes.

Acknowledgements. The research was supported by the National Science Foundation (DRL-1661036, 1713110; LDI-1934745), the Wisconsin Alumni Research Foundation, and the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin-Madison. The opinions, findings, and conclusions do not reflect the views of the funding agencies, cooperating institutions, or other individuals.

References

1. Anderson, J.R., Corbett, A.T., Koedinger K.R., Pelletier, R.: Cognitive tutors: lessons learned. *J. Learn. Sci.* (1995). https://doi.org/10.1207/s15327809jls0402_2
2. Arastoopour, G.I.: Connected design rationale: modeling and measuring engineering design learning. Unpublished Doctoral Dissertation. University of Wisconsin-Madison (2017)
3. Bakharia, A.: On the equivalence of inductive content analysis and topic modeling. In: Eagan, B., Misfeldt, M., Siebert-Evenstone, A. (eds.) ICQE 2019. CCIS, vol. 1112, pp. 291–298. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33232-7_25
4. Baumer, E.P.S., Mimno, D., Guha, S., Quan, E., Gay, G.K.: Comparing grounded theory and topic modeling: extreme divergence or unlikely convergence? *J. Assoc. Inf. Sci. Technol.* **68**(6), 1397–1410 (2017). <https://doi.org/10.1002/asi.23786>
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
6. Cai, Z., Graesser, A.C., Hu, X.: ASAT: AutoTutor script authoring tool. In: Sottolare, R., Graesser, A.C., Hu, X., Brawner, K. (eds.) *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools*, pp. 199–210. Army Research Laboratory, Orlando (2015)
7. Cai, Z., Li, H., Hu, X., Graesser, A.C.: Can word probabilities from LDA be simply added up to represent documents? In: *Proceedings of the 9th International Conference on Educational Data Mining*, pp. 577–578 (2016)
8. Cai, Z., Siebert-Evenstone, A., Eagan, B., Shaffer, D.W., Hu, X., Graesser, A.C.: nCoder+: a semantic tool for improving recall of nCoder coding. In: Eagan, B., Misfeldt, M., Siebert-Evenstone, A. (eds.) ICQE 2019. CCIS, vol. 1112, pp. 41–54. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33232-7_4
9. Cai, Z., et al.: Trialog in ARIES: user input assessment in an intelligent tutoring system. In: *Proceedings of the 3rd IEEE International Conference on Intelligent Computing and Intelligent Systems*, pp. 429–433 (2010). <https://doi.org/10.13140/2.1.4284.5446>
10. Charmaz, K.: *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. SAGE, Thousand Oaks (2006)
11. Chen, N.: Challenges of applying machine learning to qualitative coding. In: *ACM SIGCHI Workshop on Human-Centered Machine Learning* (2016)
12. Chesler, N.C., Ruis, A.R., Collier, W., Swiecki, Z., Arastoopour, G., Shaffer, D.W.: A novel paradigm for engineering education: virtual internships with individualized mentoring and assessment of engineering thinking. *J. Biomech. Eng.* **137**(2), 1–8 (2015). <https://doi.org/10.1115/1.4029235>

13. D'Angelo, C., Arastoopour, G., Chesler, N., Shaffer, D.W.: Collaborating in a virtual engineering internship. In: *Connecting Computer-Supported Collaborative Learning to Policy and Practice: CSCL 2011 Conference Proceedings - Short Papers and Posters, 9th International Computer-Supported Collaborative Learning Conference* (2011)
14. Dowell, N.M., et al.: Modeling learners' social centrality and performance through language and discourse. In: *Educational Data Mining – EDM 2015*, pp. 250–257 (2015)
15. Eagan, B.R., Serlin, R., Ruis, A., Arastoopour, G., Shaffer, D.W.: Can we rely on IRR? Testing the assumptions of inter-rater reliability. In: *CSCL 2017 Proceedings, Cim*, pp. 529–532 (2017)
16. Eagan, B.R., Swiecki, Z., Farrell, C., Shaffer, D.W.: The binary replicate test: determining the sensitivity of CSCL models to coding error. In: *Computer-Supported Collaborative Learning Conference, CSCL* (2019)
17. Geertz, C.: *The Interpretation of Cultures*. Basic Books, New York (1973)
18. Glaser, B.G., Strauss, A.L.: *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, New York (1967)
19. Graesser, A.C.: Conversations with AutoTutor help students learn. *Int. J. Artif. Intell. Educ.* **26**(1), 124–132 (2016). <https://doi.org/10.1007/s40593-015-0086-4>
20. Grün, B., Hornik, K.: Topicmodels: an R package for fitting topic models. *J. Stat. Softw.* (2011). <https://doi.org/10.18637/jss.v040.i13>
21. Hardy, M.: Career Interview with Ian Shaw. *Qualitative Social Work*. (2019). <https://doi.org/10.1177/1473325017727342>
22. Liu, M., et al.: Understanding MOOCs as an emerging online learning tool: perspectives from the students. *Am. J. Dist. Educ.* (2014). <https://doi.org/10.1080/08923647.2014.926145>
23. Mayfield, E., Adamson, D., Rosé, C.P.: *LightSide Researcher's Workbench* (Version 2.1.2)[Computer Software]. LightSide, Pittsburgh (2013)
24. Miles, M.B., Huberman, A.M.: *Qualitative Data Analysis* (Second Edition) (1994)
25. Ngulube, P.: Qualitative data analysis and interpretation: systematic search for meaning. In: *Addressing Research Challenges: Making Headway for Developing Researchers* (2015)
26. Nikolenko, S.I., Koltsov, S., Koltsova, O.: Topic modeling for qualitative studies. *J. Inf. Sci.* 1–15 (2015). <https://doi.org/10.1177/0165551515617393>
27. Peters, G., Zörgö, S.: Introduction to the Reproducible Open Coding Kit (ROCK). *Psyarxiv* (2019). <https://doi.org/10.31234/osf.io/stex9>
28. Rezaei, E., Zavaraki, E.Z., Hatami, J., Abadi, K.A., Delavar, A.: The effect of MOOCs instructional design model based on students' learning and motivation. *Man in India*. **97**, 115–126 (2017)
29. Miles, M.B., Huberman, A.M., Saldana, J.: *Qualitative Data Analysis: A Methods Sourcebook*. SAGE, Thousand Oaks (2019)
30. Ruis, A.R., Rosser, A.A., Nathwani, J.N., Beems, M.V., Jung, S.A., Pugh, C.M.: Multiple uses for procedural simulators in continuing medical education contexts. In: Eagan, B., Misfeldt, M., Siebert-Evenstone, A. (eds.) *ICQE 2019. CCIS*, vol. 1112, pp. 211–222. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33232-7_18
31. Snowdon, P.F.: What Is Le Penseur Really Doing? In: Dolby, D. (ed.) *Ryle on Mind and Language*. PD, pp. 116–125. Palgrave Macmillan UK, London (2014). https://doi.org/10.1057/9781137476203_7
32. Shaffer, D.W.: *Quantitative Ethnography*. Cathcart Press, Madison (2017)
33. Strauss, A., Corbin, J.: Basics of qualitative research: techniques and grounded theory procedures for developing grounded theory. (1998). <https://doi.org/10.2307/328955>
34. Swiecki, Z., Ruis, A.R., Gautam, D., Rus, V., Shaffer, D.W.: Understanding when students are active-in-thinking through modeling-in-context. *Br. J. Edu. Technol.* (2019). <https://doi.org/10.1111/bjet.12869>

35. Theelen, H., Willems, M.C., van den Beemt, A., Conijn, R., den Brok, P.: Virtual internships in blended environments to prepare preservice teachers for the professional teaching context. *Br. J. Edu. Technol.* (2020). <https://doi.org/10.1111/bjet.12760>
36. Wang, Y., Baker, R.: Content or platform: why do students complete MOOCs? *J. Online Learn. Teach.* (2015)
37. Wang, Y., Baker, R.: Grit and Intention: why do learners complete MOOCs? *Int. Rev. Res. Open Dist. Learn.* (2018). <https://doi.org/10.19173/irrodl.v19i3.3393>
38. Yousef, A.M.F., Chatti, M.A., Schroeder, Ul, Wosnitza, M., Jakobs, H.: MOOCs a review of the state-of-the-art. In: *Proceedings of the 6th International Conference on Computer Supported Education – CSEDU 2014*, pp. 9–20 (2014)