An Automated Approach to Assessing an Application Tutorial's Difficulty

Shahed Anzarus Sabab[1], Adnan Khan[1], Parmit K. Chilana[2], Joanna McGrenere[3], Andrea Bunt[1]
[1]*Department of Computer Science* *University of Manitoba* Winnipeg, MB, Canada {sabab05, akhan, bunt}@cs.umanitoba.ca
[2]*Department of Computing Science* *Simon Fraser University* Burnaby, BC, Canada pchilana@cs.sfu.ca
[3]*Department of Computer Science* *University of British Columbia* Vancouver, BC, Canada joanna@cs.ubc.ca

***Abstract*— Online step-by-step text and video tutorials play an integral role in learning feature-rich software applications. However, when searching, users can find it difficult to assess whether a tutorial is designed for their level of software expertise. Novice users can struggle when a tutorial is out of their reach, whereas more advanced users can end up wasting time with overly simple, first-principles instruction. To assist users in selecting tutorials, we investigate the feasibility of using machine-learning techniques to automatically assess a tutorial's difficulty. Using Photoshop as our primary testbed, we develop a set of distinguishable tutorial features, and use these features to train a classifier that can label a tutorial as either Beginner or Advanced with 85% accuracy. To illustrate a potential application, we developed a tutorial browsing interface called *TutVis*. Our initial user evaluation provides insight into *TutVis's* ability to support users in a range of tutorial selection scenarios.**

Keywords—software learnability, topic modeling, expertise.

I. Introduction

Online tutorials are among the most popular and heavily used resources for learning and using feature-rich software applications, such as AutoCAD, Photoshop, Fusion360, and many others [4,39]. There is an abundance of tutorials online (e.g., over 28,160 video and text tutorials on the popular aggregator site tutsplus.com) and, in comparison to software forums or Q&A sites, tutorials often cover full workflows, illustrating the step-by-step progression of a task.

Despite the growing popularity of online tutorials, it can be difficult for many users to locate and identify tutorials that are appropriate for their level of software expertise [12,25,63]. Expert users, for example, often desire advanced tutorials that cover novel tips and techniques [20,28,35] and contain compact workflow representations. However, such tutorials often assume certain software skills and knowledge of the application's vocabulary [14,20]. When a novice user tries to follow tutorials with this assumed knowledge, they can experience cognitive overload [40,48], frustration [38], and limited task success [28].

A key problem with existing online tutorials is that they often fail to provide expertise or difficulty information to help a user select an appropriate tutorial. For example, when sampling from around 8,000 Photoshop tutorials on tutsplus.com, we found that only 8% provided any difficulty information. This led us to ask: Can we create technology that automatically classifies a tutorial's difficulty? If so, how can we present this difficulty information to users to help them select tutorials?

Given the highly structured nature of many feature-rich tutorials, with their step-based [39], and command-oriented workflows [31], we use a machine-learning approach to uncover properties of advanced vs. beginner tutorials. Using Photoshop as our initial testbed, we identify and engineer a set of tutorial features that we extract from a tutorial's text (or captions in the case of video tutorials) including: topics, length, text complexity, and the density of command references. We then investigate the impact of these features on classification accuracy. Specifically, we train a model using 750 tutorials with existing difficulty labels (obtained from 9 online tutorial repositories) using different feature combinations. Using 10-fold cross-validation, we show that our best model achieves an accuracy of 85% when classifying tutorials as either beginner or advanced. We demonstrate some degree of generalizability of our approach and feature sets by applying them to a 3D modeling application (Fusion360).

To illustrate a user-centered application of our classifier, we created a prototype tutorial browsing interface called *TutVis* (Fig 4). *TutVis* supports tutorial selection by annotating each tutorial with its automatically generated difficulty label, along with interface components that summarize other tutorial features (i.e., those leveraged by our classifier). In a proof-of-concept evaluation with 12 participants, we compared *TutVis* to two other interfaces that displayed subsets of the annotations (e.g., only the difficulty labels). Our results suggest that participants prefer having information on both the tutorial's difficulty level and the high-level task covered and that this combination of information helps increase their tutorial selection confidence.

To summarize our contributions: 1) We identify and investigate features (e.g., topic, length) that differentiate feature-rich software tutorials that are appropriate for experts from those for beginners. 2) We illustrate that a machine-learning model can leverage these features for an 85% classification accuracy. 3) We show how the classifier's decision and its features can be presented through our *TutVis* system. 4) We provide initial insights from a proof-of-concept evaluation on how *TutVis* impacts tutorial selection tasks. Our work presents an initial step towards automatic detection of a tutorial's difficulty and has implications for designing systems that aim to support users in selecting appropriate learning resources for feature-rich software applications.

II. Related Work

A. Characterizing and Classifying Software Expertise

Prior work has recognized the wide range of expertise that users bring to their interactions with feature-rich software. Building on Nielsen's categorization of general user interface expertise [47], Grossman et al. define feature-rich software

978-1-7281-6901-9/20/$31.00 ©2020 IEEE

expertise according to the following dimensions: experience with computers, experience with the software's interface, domain knowledge and experience with similar software [20].

Prior research has looked at the feasibility of automatically detecting software expertise, a key step for supporting users of differing skill levels. One area of focus has been on capturing and analyzing low-level interface operations. Examples of such expertise indicators include: the time to perform commands [19], the rate of interface actions [24], pauses or dwells [50], mouse motions [17], and menu access times [25]. Our work aims to accommodate different skill levels by automatically assessing the difficulty of tutorials available online.

Research has also investigated how users' expertise affects their use of an application's command set. Lawson et al.'s study of spreadsheet use found expertise-related workflow differences [35]. Matejka et al. found that command usage frequency can be an indicator of software expertise [37]. We leverage these findings to investigate command-oriented tutorial features that serve to discriminate between beginner and advanced tutorials.

B. Improving the Usability of Software Tutorials

Many software users, especially beginners often struggle in locating a relevant tutorial for a given task [28]. Given the important role of tutorials in software learning, a wide body of work has looked at how to support tutorial use and retrieval.

In supporting tutorial use, one approach has been to integrate tutorials with the target applications, for example, through overlays that help users find tutorial commands [26], or techniques that use application context to control a video tutorial's progression [50]. Other approaches include using automation to reduce tutorial workload [31], adding gamification elements [36], and augmenting tutorials with input from the user community [7,33,51].

Some prior approaches have explored annotating software tutorials to make it easier for users to select, appraise, and navigate them. Examples of previously explored tutorial annotations include: commands covered [15,49], UI events [2,21], other users' viewing patterns [29], and the location of workflow steps within a video [30,62]. This prior work has leveraged a mix of automated (e.g., [15,49,51]) and crowdsourcing techniques (e.g., [30]) to create the annotations.

Despite all the research in improving user interaction with tutorials, there is little prior work on providing users with information about the difficulty level of the application content covered. One exception is Social CheatSheet [59], a system for creating and sharing software instructions and tutorials, which proposed a social voting mechanism to classify an instruction set's difficulty level. Also highly relevant to our work is Wang et al.'s work on identifying tutorial tasks [61]. Their approach leveraged command usage logs and topic modeling to identify latent tutorial topics. They then had experts manually assign human-readable topic labels, consisting of the task covered and its difficulty. Our work differs in that we use machine learning to classify a tutorial's difficulty level automatically. Our approach also does not require access to usage logs. Finally, our work provides insights into how tutorial difficulty information can affect users' tutorial selection tasks.

III. Method Overview

Our goal is to investigate the feasibility of using machine learning to automatically label an application tutorial's difficulty. In this section, we describe the data that we collected for classifier training, our data preprocessing, our feature definition and investigation process, and the statistically significant differences between advanced vs. beginner tutorials based on our features.

A. Collecting Labeled Photoshop Tutorials

We began by collecting a corpus of already labeled tutorials for use as ground truth for classifier training and testing. Our initial investigation is confined to Photoshop tutorials as it is a widely used application, frequently studied in feature-rich software research [7,8,11,30,55]. We explore our approach's generalizability to 3D modeling software in Section IV.C.

To ensure high-quality difficulty labels, we consulted only tutorial sources that appeared to have a strict editorial process or accepted tutorials from only experienced authors. Our final sample contained tutorials from 9 sources: envatotuts+ (70.3%), Photoshop Star (9.5%), Adobe (7.2%), Creative Bloq (3.7%), tutpad (3.1%), tutvid (2.7%), Pelfusion (1.5%), PSD Vault (1.3%), and 99 designs (0.8%). As a proof-of-concept, we focused on building a classifier to distinguish between two difficulty levels, a choice motivated by the fact that six of our sources used a similar two-level scheme (e.g., "Advanced / Beginner"). The remaining three sources used three difficulty levels (e.g., "Advanced / Intermediate / Beginner"). For these sources, we labeled both the "Intermediate" and "Advanced" tutorials as "Advanced" in our corpus.

Our final corpus had 750 tutorials (i.e., 375 advanced and 375 beginner), with equal distributions of video and text tutorials across each difficulty level (70% text and 30% video tutorials).

B. Data Preprocessing

Our next step was data preprocessing, a common step in classification to remove known sources of noise [58]. We focus our investigation on textual features only, which in the case of video tutorials, came from the transcripts. Guided by related works [44,57] and informal experimentations, we performed four preprocessing steps: 1) We converted all tutorial text and video transcripts into lowercase, and divided the text into tokens (i.e., small pieces or words). 2) We removed special characters, articles, punctuation, numerals, prepositions, conjunctions, pronouns, and stopwords. 3) We converted words into their base forms (known as lemmatization [1]). 4) We created word bigrams [6], consisting of frequently co-occurring words.

C. Feature Investigation

After preprocessing, we created a set of potential features to train our classifier. Based on prior works on software expertise and learnability (e.g., [19,20,35,41,59]) and conducting our own informal feature investigations, we settled on: topics, command ratio, word repetition, text complexity, and length. We briefly discuss our motivation for each feature, and how we developed the features from the tutorial text/transcript.

1) Tutorial Topics: Prior work suggests a potential relationship between a tutorial's higher-level topic and its difficulty level. For example, user comments posted to online

tutorials describe certain tutorials as covering expert techniques [32]. In Wang et al.'s work on identifying tutorial tasks via command usage logs, when they asked experts to provide human-readable labels for their machine-generated topics, the labels included both task and difficulty information [61].

Inspired by this prior work, we used topic modeling to associate each tutorial with a set of topics that it covers. We then leverage these topic models (i.e., document-topic distributions) in classifying each tutorial's difficulty. Due to its ability to capture the hidden structure of text [46,61], we used Latent Dirichlet Allocation (LDA) [3]. LDA is an unsupervised topic-modeling technique that assumes each document (i.e., tutorial in our case) is a mixture of topics, each of which is present in different proportions. These latent topics essentially represent clusters of commonly occurring words.

We generated two different LDA topic models: 1) a *Topics-All* model. which considered all of the preprocessed text; and 2) a *Topic-Commands* model, which considered only command references. For our *Topic-Commands* model, we applied techniques from prior work on automatically identifying direct and indirect command references from tutorials [15,49] to create a Photoshop command dictionary. We added 1096 direct command references to our dictionary using a Photoshop option that lists all commands (i.e., Edit > Keyboard Shortcut > Summarize). To collect examples of indirect references, we manually annotated a subset of 70 Photoshop tutorials (35 Advanced and 35 Beginner). During this process, we looked for colloquial forms of the direct commands (e.g., an indirect reference of "set blending mode" is "adjust the blending mode"). We added an additional 2470 indirect command references to our dictionary via this hand-annotation approach.

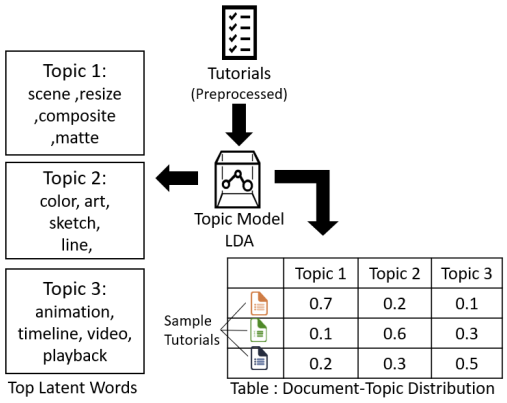


Fig. 1. An example of LDA topic model output for 3 sample topics.

We used both sources of text (all preprocessed text and only command references) as input to LDA. LDA produces a document-topic distribution matrix as output (see Fig 1. for an example), which we used as features in the classification. We generated 30 LDA topics, based on a topic evaluation metric called the topic coherence score [43,45,60].

2) Command Ratio (CR): Matejka et al. found a connection between a user's expertise level and the frequency in which they used different commands [37]. To investigate whether advanced tutorials make heavier usage of commands than beginner tutorials, we used our command dictionary to count the number of command references. To account for tutorial length, we used a tutorial's command ratio (CR), which represents the percentage of words in the tutorial that refer to a Photoshop command.

3) Word Repetition (WR): Our informal exploration of tutorials suggested that advanced tutorials tend to focus on specific effects or tasks (e.g., "Creating a Sketch Effect") whereas the beginner tutorials were often broader (e.g., "Demonstrating Different Retouching Tools"). We created a word repetition feature based on a speculation that there might be greater repetition in advanced tutorials owing to their more focused nature. We defined this feature as:

$$\text{Repeated Words (WR)} = \frac{Number\ of\ repeated\ words}{Number\ of\ total\ words} \times 100$$

4) Text Complexity (TC): Based on our informal investigation, we also speculated that advanced tutorials might use more complex language. To capture this, we used a consensus score of 7 different formulas as advocated in prior work [13] (i.e., Flesch Reading Ease, Flesch-Kincaid Grade Level, Fog Scale, SMOG Index, Coleman-Liau Index, Automatic Readability Index, Linsear Write Formula). The text complexity score ranges from 1-12, with higher values for more complex text.

5) Tutorial Length (Len): Finally, our informal investigation suggested that advanced tutorials tended to be lengthier than beginner tutorials. We represent tutorial length as the number of words present (i.e., word count).

D. Differences between Advanced vs. Beginner Tutorials

For features that we could summarize using means (e.g., command ratio, length, word repetition, and text complexity), we looked for significant differences between the advanced and beginner tutorials in our dataset (using 2-tailed Independent T-tests). Table I shows that advanced tutorials are significantly longer and have more repeated words than beginner tutorials. Contrary to our speculation, beginner tutorials use more complex language (according to the readability measures); however, the effect size (Cohen's d) is small. We did not find a significant difference in the density of command references (i.e., command ratio) between advanced and beginner tutorials.

TABLE I. DIFFERENCES BETWEEN ADVANCED AND BEGINNER TUTORIALS FOR FOUR OF OUR FEATURES.

	Adv. Mean(s.d.)	Beg. Mean(s.d.)	Sig	Cohen's d
CR	33.3 (10)	34.3 (11.2)	p = 0.10	0.1
Length	2275.8 (1124.1)	1461 (841.8)	**p < 0.001**	0.8
WR	71.7 (8.5)	68 (7.9)	**p < 0.001**	0.5
TC	7.5 (1.6)	8.1 (1.7)	**p < 0.001**	0.4

CR: Command Ratio, TC: Text Complexity, WR: Word Repetition, Len: Length

IV. MODEL GENERATION AND PERFORMANCE ANALYSIS

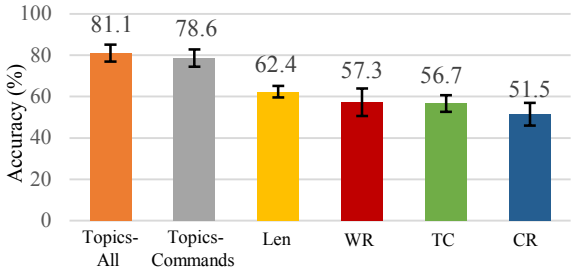
In this section, we investigate the feasibility of automatically classifying a tutorial as advanced or beginner by examining the

performance of different possible classifiers. We also examine the discriminatory power of our different features.

Due to its robustness and that it tends to be less prone to overfitting than some other approaches (e.g., Decision Tree, Naïve Bayes), we used Random Forest for the classification [5]. We optimized classifier parameters using Grid Search [53]. To evaluate each classifier's performance, we use a standard cross-validation approach, with 10 folds (using StratifiedKFold [65]). In other words, each classifier (aka model) was trained and validated through 10 trials, where each trial used a different 90% of the data as training samples and the remaining 10% of the data as testing samples. Because of our balanced dataset, we report accuracy as our performance metric.

A. *Impact of Individual Feature Sets on Classifier Accuracy*

We initially investigated the impact of the individual feature sets (topics, length, word repetition, text complexity, and command ratio) on classifier performance. As a reminder, we have two topic models: *Topics-All* and *Topics-Commands.*



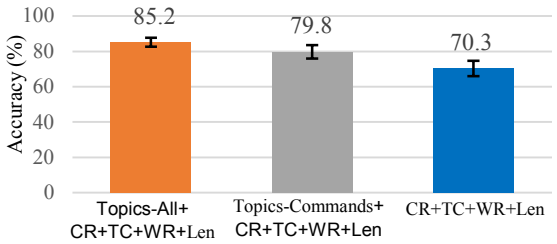
CR: Command Ratio, TC: Text Complexity, WR: Word Repetition, Len: Length

Fig. 2. Model performance using individual features. Error bars represent standard deviation.

Fig 2. shows that our classifier achieved the best performance (i.e., *accuracy = 81.1%*) when it was trained using the topics derived from all of the text. Accuracy dropped slightly (to *78.6%*) when considering only the command references. In other words, topics are our most informative feature, and the difficulty information is not only confined to the Photoshop command references. Conversely, command ratio was the least informative feature, resulting in baseline accuracy (i.e., *50%* in this 2-class classification problem). The models trained with the other feature sets (text complexity, word repetition, and length) also did not perform well. Thus, while there are significant differences in mean values for these tutorial features, these differences were not strong enough to distinguish between advanced and beginner tutorials.

B. *Impact of Combining Feature Sets on Classifier Accuracy*

We also investigated the impact of combining different features on classifier accuracy. Fig. 3 shows that our classifier performed best (accuracy = 85.2%, F1 = 0.85, AUC = 0.86, Kappa = 0.71) when we included all of our features. In this highest-performing model, the topics were derived from all of the text. Accuracy dropped slightly (to 79.8%) when using the command-only topic distributions. These results indicate that while some of our features lack discriminatory power when used in isolation (see Fig. 2), they performed better when used in combination.



CR: Command Ratio, TC: Text Complexity, WR: Word Repetition, Len: Length

Fig. 3. Model performance using combined features. Error bars represent standard deviation.

C. *Generalizing to 3D Modeling Tutorials*

To investigate the generalizability of our features, we evaluated our best model's performance (CR, TC, WR, Len, and Topics-All) using tutorials for a different feature-rich application: 3D modeling software. For this purpose, we collected 210 labeled tutorials for the application Fusion 360 (Advanced 105, Beginner 105, 90% video tutorials) and constructed a Fusion 360 command dictionary. Our data preprocessing and feature engineering procedures were identical to those described in Section III, with the exception that we had LDA produce 20 topics (guided again by the topic coherence score). With this dataset, our classifier achieved an average of 81.4% accuracy (s.d.=9.2) when trained/tested using 10-fold cross-validation. This provides encouraging initial evidence that our feature sets, and classification techniques generalize beyond Photoshop to other kinds of feature-rich software.

V. Topic Interpretation and Labeling

Our model performance analysis revealed that topic distribution (generated via LDA) is our most informative tutorial feature. As a reminder, LDA generates latent words for each topic and applies a generic label (e.g., "Topic 1", "Topic 2" in Fig. 1). We were interested in whether these topics have meaningful application-level semantics that, for example, a system could present to users along with the difficulty assessment. In this section, we briefly describe how we went from this LDA output to the human-readable labels that we used in our *TutVis* system. Further details can be found in [56].

Labels for machine-generated topics can be assigned by humans manually [54,61] or through automated techniques [34,42]. However, human-generated labels often give users more insights into the nature of the topics [23]. One manual approach is to have domain experts assign labels using top latent words from the topic-word distribution table (e.g., "scene", "resize", "composite", and "matte" in Fig. 1) that LDA generates automatically [61]. When we tried this approach, we found that the latent words included a number of generic Photoshop terms (such as scene, matte, animation, timeline) that made it difficult to associate them with distinct high-level topics.

We instead manually inspected several tutorials per topic, noting similarities and differences in the high-level tasks that they covered. Based on this analysis, we found that about 30% of the machine-identified topics seemed to represent clear high-level Photoshop tasks, and we found crafting labels for these relatively straightforward. For other LDA topics, we saw clear tasks within the topic, but did not see enough semantic

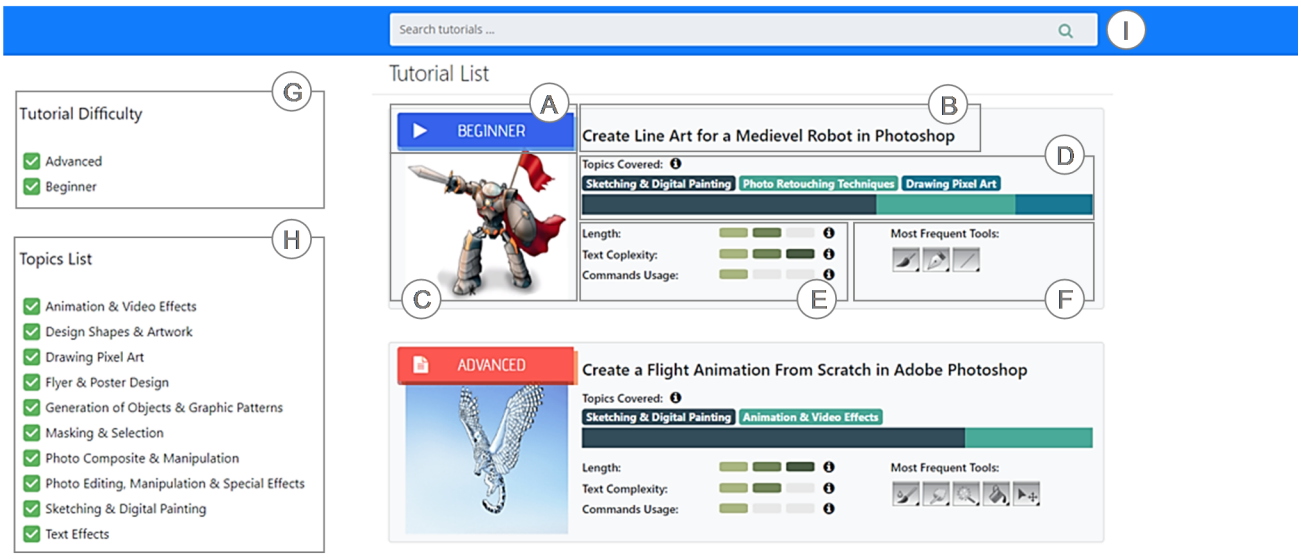


Fig. 4. The *TutVis* interface, which presents a list of tutorials with difficulty (A), title (B), thumbnail image (C), topics covered (D), length, text complexity, commands usage (E) and most frequently used tools (F). *TutVis* also provides filtering options (G, H) and a search bar (I).

differences relative to some other topics to warrant unique labels. For these (about 60% of the topics), we grouped subsets of the topics together and assigned a common label. For example, we assigned the label *photo composite and manipulation* to 5 different LDA topics, all of which involved manipulating photos and creating a hypothetical or surreal scenery by combining the manipulated photos. Finally, a handful of the topics (about 10%) covered heterogeneous tutorials. We handled these cases by labeling them generically according to their commonalities (e.g., *editing & selection)*.

In the end, we created 18 topic labels to cover the 30 topics generated by LDA. We asked an additional Photoshop expert to verify the semantics of our labels. The expert suggested some minor wording adjustments, but otherwise felt that our labels provided reasonable descriptions of the high-level tasks covered. Thus overall, we found that the vast majority of the LDA topics did have meaningful Photoshop semantics that we could use to assign human-readable labels manually.

VI. Tutorial Selection Interface

To illustrate how our classifier and its features could be used to help users select tutorials, we developed the *TutVis* prototype. As shown in Fig. 4, *TutVis* uses our classifier to annotate each tutorial with an automatically generated difficulty assessment. *TutVis* also summarizes other features that contributed to this difficulty assessment through interface components representing: the topics covered, the text complexity, the length, and commands usage (renamed from command ratio in section III.C.2 based on pilot testing). We refined the visual representations of these features iteratively based on pilot testing. For topics, we chose to include only those which contributed at least 10% to the tutorial's overall topic distribution, resulting in tutorials having at most three topics listed (Fig 4., D, the stack bar shows the distribution of the topics). We did not include our model's word repetition feature after pilot testing with different visual representations revealed that users found this feature difficult to understand.

Building on prior work on command-oriented tutorial browsing interfaces [31,49], *TutVis* also lists the tools that are used most frequently in the tutorial, as well as the tutorial's title and final image (i.e., thumbnail). Users can click on a tutorial for a more detailed view and can hover to obtain more information on the different interface components. *TutVis* also allows users to filter tutorials according to topic and difficulty, or to search using keywords from the titles or the topics.

VII. Tutorial Selection Study

We conducted an initial user study to evaluate *TutVis's* utility. We aimed primarily to gain qualitative insight into the value of the difficulty labels in helping learners select a tutorial from a tutorial repository, as well as *TutVis's* representations of the different tutorial features (i.e., topics, length, text complexity, commands usage).

A. Participants

We recruited 12 participants (8 male, 4 female) through advertisements posted on a local university campus, social media, and word of mouth. All participants were familiar with Photoshop: 5 self-reported as beginners (i.e., use Photoshop once a month or less), 5 as intermediates (i.e., use Photoshop at least once a week), 2 as experts (i.e., use Photoshop daily). Participants received $20 for their participation.

B. Study Conditions and Tutorials

Our study had a within-subjects design with three conditions, each with a different tutorial browsing interface (*Baseline*, *TutDiff,* and *TutVis*). The three conditions differed in the number of tutorial features that were displayed as follows:

1. ***Baseline:*** each tutorial was annotated with the title, thumbnail image, and most frequently used tools (i.e., Fig. 4: B, C, F).
2. ***TutDiff*:** all information in the *Baseline* interface plus the auto-generated difficulty labels (advanced/beginner) (i.e., Fig. 4: A, B, C, F, G).

3. ***TutVis*:** our complete *TutVis* system (see Section VI). The additional annotations available in this condition included: topics, length, text complexity, and commands usage (i.e., Fig. 4; D, E).

In studying three conditions, we wanted to observe how participants make use of the annotations available to them to inform their tutorial selections. We were primarily interested in using structured observational data to gain qualitative insight into the value of the different interface components. We included the *Baseline* condition to give participants experience with something representative of a "status-quo". *TutDiff* is an intermediary condition, which we included to examine the value of automated difficulty classification over the status quo.

Each tutorial browsing interface contained 50 Photoshop tutorials. We selected three mutually exclusive sets of varied tutorials from our tutorial corpus (in terms of topics, difficulty, length, etc.), which we randomly assigned to each condition. To replicate our model's overall performance (85% accuracy), each set had 7 tutorials with incorrect difficulty labels (i.e., misclassified as advanced or beginner). We fully counterbalanced interface order across participants.

C. Procedure

After completing the demographic questionnaire, participants completed three tutorial selection tasks per condition (i.e., nine in total). Each selection task presented a different scenario and asked the participants to find a tutorial accordingly. Our scenarios were motivated by previous research on the different reasons that users search for tutorials online (e.g., [11,32]). The first focused on a scenario with a sense of urgency, the second involved an exploratory search, and the third focused on wanting a tutorial of particular difficulty. We created three isomorphic scenarios sets, which we iteratively refined and pilot tested. Table II shows one of the scenario sets.

TABLE II. ONE SET OF TUTORIAL SELECTION SCENARIO

Task	Task Description
Sense of urgency (1st Task)	You are assigned the task of creating an advertisement for a fundraising occasion. You want to complete this task quickly. Select a tutorial that you think would serve as the best starting point for you.
Exploratory search (2nd Task)	You are free for the whole afternoon and you are interested in learning about digital drawing. Find a tutorial that would give you some insight into digital drawing.
Sense of difficulty (3rd Task)	You have a friend who has never used Photoshop. Recently, he asked you for help in finding tutorials on how to change an image background. Find a suitable tutorial for your friend.

A challenge in studying tutorial browsing interfaces for an application like Photoshop tutorials is that the tutorials tend to take a significant amount of time to complete (e.g., 30-90 mins per tutorial). Therefore, to focus the study time on tutorial selection data, we asked participants to spend around 7-10 minutes per selection task but did not require them to complete their selected tutorial. This follows previously established methodology for evaluating tutorial selection interfaces [31].

As indicated earlier, our primary goal was to qualitatively assess which interface components factored into participants' tutorial selections. A secondary goal was to quantitatively assess participants' self-reported confidence that their selections were appropriate for the given scenarios. With these goals in mind, we asked participants to think-aloud while browsing tutorials, and we also recorded eye gaze information using a Tobii Eye Tracker 4C. After each condition, participants completed a short questionnaire to report directly on: i) the interface components they used to guide their tutorial selections and ii) their level of confidence in their selections (using a 5-pt Likert scale). At the end of the session, we conducted a semi-structured interview, where we asked participants about their preferences and how they used the different components. Each session lasted approximately 1.5 hours.

D. Results: Tutorial Selection Study

1) Preferences and Confidence Levels: In the interview, we asked participants to rank the three interfaces according to their preferences. All 12 participants ranked *TutVis* as their most preferred interface. The *Baseline* condition had little support, with 11/12 participants rating it as their least preferred.

We also compared participants' tutorial selection confidence levels (as reported on a 5-pt Likert scale) using Friedman's two-way ANOVA. We found a significant main effect of browsing interface on selection confidence ($\chi^2(2) = 11.267$, $p = 0.004$). Posthoc comparisons (Bonferroni adjusted) indicated that participants felt more confident when using *TutVis* (mean = 4.7, s.d. = 0.5) than when using *Baseline* (mean = 3.6, s.d. = 0.8, $p = 0.006$). There were also trends suggesting that participants were more confident with *TutVis* than with *TutDiff* (mean = 4.1, s.d. = 0.9, $p = 0.068$), and that they were more confident with *TutDiff* than with *Baseline* ($p = 0.084$).

2) Impact of Interface Conditions on Tutorial Exploration: We analyzed tutorial exploration behavior using Friedman's two-way ANOVA and found significant main effects of browsing interface on the number of tutorials inspected ($\chi^2(2) = 7.762$, $p = 0.021$) and the time spent browsing ($\chi^2(2) = 8.773$, $p = 0.012$). Posthoc comparions revealed that participants inspected fewer tutorials with TutVis (mean = 3.1 tutorials) than with Baseline (mean = 5.2 tutorials; $p = 0.032$). Similarly, participants spent less time browsing with TutVis (mean = 10.4 mins) than with Baseline (mean = 13.5 mins, $p = 0.032$).

3) Individual Component Usage: The post-condition questionnaire asked participants to indicate which interface components they had used during that condition. For compactness, Fig. 5 presents data from all three conditions; however, as a reminder, not all features were available in each condition (see section *B.Study Conditions*). When the difficulty labels were present (in *TutVis* and *TutDiff*), the majority of participants reported using them, particularly with *TutDiff* (11/12 participants). Of *TutVis*'s additional components, the topics were the most heavily used - all participants reported using them. The availability of the topic labels seemed to decrease reliance on the difficulty labels somewhat, with 9/12 participants using them in *TutVis*. This suggests that for 3/12

participants, some of the information that they were looking for with the difficulty labels (in *TutDiff*) was potentially better captured by the topic labels. The remaining 9/12 continued to use the difficulty labels even with the topic information available. Participants reported using title and thumbnail in all conditions, but less so with *TutVis*, where some participants seemed to rely on the topic labels instead to assess emphasis. Other components (e.g., length, text complexity, commands usage, and frequent tools) were not as heavily reported.

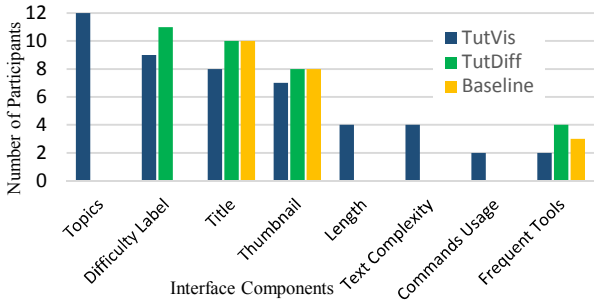


Fig. 5. Self-reported interface components used.

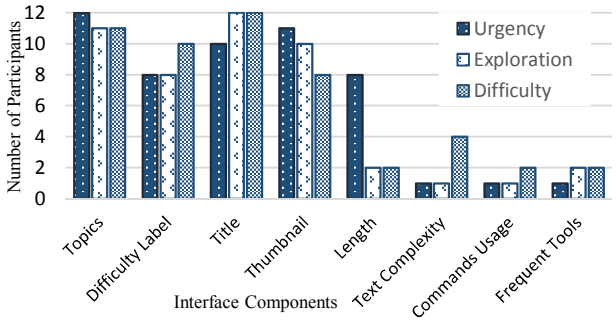


Fig. 6. Interface components used in the different task scenarios according eye-gaze and think-aloud data (in *TutVis* only).

To provide further insight into how each selection scenario impacted interface component usage, we turned to the think-aloud transcripts and the eye-gaze data. To analyze the gaze data, we leveraged heatmaps generated by a software extension of Tobii [66]. We considered only those components with the longest fixation duration as determined by the application (i.e., dwells of at least 2.2 ms, guided by [10]). Following previous work on combining eye-gaze and think-aloud data [9], we retained only the fixations where the participant also mentioned using the component to guide their selection. We conducted this analysis on the *TutVis* data only, since this condition contains all interface components. Fig. 6 shows that while there was some variation in component usage across tasks, there were no dramatic differences. The one notable exception is the length component, which was used by 8/12 participants in the task that conveyed a sense of urgency, and by only 2/12 participants in the other tasks. Fig. 6 also shows that the majority of participants used the difficulty labels in all three tasks, as opposed to in only the task that emphasized the expertise of the target user. Finally, the figure indicates heavier reliance on the titles and thumbnails than participants reported in the post-condition questionnaires.

4) Perspectives on TutVis's Components: Our semi-structured interviews provided further insight into participants' perspectives on *TutVis's* unique components relative to the *Baseline* condition.

All participants were enthusiastic about the topic information, a primary reason being that the topic labels tended to be more useful/accurate than the title in summarizing the tutorial's emphasis. One participant felt the topics served a similar function as the preface of a book:

> *"It is giving you a type of outline […]. It is like a preface to a book.." – (P10)*

Many participants found the expertise labels to be a particularly useful way to streamline the list of tutorials to only those that would match the desired expertise level:

> *"He is from a different background [beginner user]. He might flip [out] if provided with more technical jargon [advanced tutorial] […]"- (P10)*

The expert participants also appreciated the difficulty labels when they wanted a tutorial that would go beyond the basics required for accomplishing a task:

> *"If I am doing this […] I vote to have something more stylish, more attractive […], eye catchy. So that's why I am choosing this [advanced tutorial]"- (P7)*

The length component was mostly used in the urgency task scenario (see Fig. 6). The expert participants indicated that they were searching for a short tutorial because they did not need in-depth explanations of the task or tool usage:

> *"[Designers] are not going to [want] videos that are like 30 minutes and that explain what selection tools are." – (P5)*

Conversely, some beginner participants were more interested in the long tutorials that show step-by-step changes:

> *"[A short tutorial] does not describe how to create a canvas. […] [This long tutorial] describes the tools you are [going to] use step-by-step […] Length is definitely helpful." – (P6)*

Thus overall, participants were most enthusiastic about *TutVis's* topics and difficulty labels. Others saw benefits in using the length component for certain tasks. Participants did not see a lot of value in the text complexity and command usage components. They felt that they could cope with various text complexity levels and found our command usage feature difficult to interpret.

5) Perspectives on Misclassifications: During the interview, we also asked participants how they would feel about misclassified difficulty labels, given our classifier's overall accuracy (85%). Participants who self-reported themselves as experts or intermediates were generally not concerned with misclassification. They felt that they either had the knowledge to further assess the tutorial before committing to it or could cope with various levels of difficulty. For example:

> *"[Following a misclassified tutorial] is not difficult for me here because I can follow each level." – (P7)*

Some worried more about misclassifications related to beginner tutorials and potential struggles in task completion:

"If I am sharing a tutorial to someone else, like a grandparent, and its actually advanced […], that's not gonna be very good." – (P1)

We reflect on this lack of symmetry in our discussion.

E. Summary of the User Study

Our results suggest value in providing users with both automatically generated difficulty labels and information on features that contribute to this classification. Our full-featured *TutVis* interface was the most preferred version. Despite inspecting fewer tutorials with *TutVis*, participants also felt more confident in their selections than they did with *Baseline*. In considering the value of the expertise labels alone, which were completely auto-generated, participants preferred *TutDiff* over *Baseline,* and the majority continued to use these labels even when provided with the topics.

VIII. Discussion

Feature-rich software users can face a number of challenges and frustrations when trying to follow a tutorial that is not written for their level of expertise [28]. Given that the majority of today's online tutorials do not provide expertise guidance, we investigated the feasibility of using machine learning to automatically generate tutorial difficulty labels. Our initial results are promising. For example, using a set of five text-based features, our classifier labeled Photoshop tutorials as either beginner or advanced with 85% accuracy. We have also demonstrated some generalizability of our overall approach and specific feature set beyond Photoshop by applying them to classify tutorials for a 3D modeling application.

The most discriminating feature in our classifier was the latent tutorial topics revealed by LDA. Further, topics derived from all of the words in the tutorial text/transcripts (*Topics-All*) led to slightly higher accuracy than topics derived from command references only (*Topics-Commands*). Many Photoshop commands can play a role in both advanced and beginner tasks, but sometimes the way the commands are used differs. Our *Topics-All* model was likely able to capture some of these usage dynamics to increase classification accuracy. Our work relies on text-based features only, opening opportunities to explore additional classification features. One idea would be to leverage advances in computer vision to generate new visual features about tutorial difficulty by analyzing objects in images and video frames [2]. Another area of future work would be to explore how using different tutorial repositories for training might impact the role of different tutorial features in classification.

Our study provides initial insight into how misclassifications might impact users. We did not observe any frustrations or confusions stemming from inaccurate labels in our think-aloud data; however, our interview data suggests that the classifier might need to be conservative when labeling a tutorial as beginner. Novice users might be more negatively impacted by a tutorial that is too advanced, becoming frustrated or discouraged. In contrast, expert users can likely leverage their existing software knowledge to detect a beginner tutorial that is mis-labeled as advanced. One way to alleviate the impact of the misclassifications would be to augment the automatically generated labels with community-based feedback about tutorial difficulty (e.g., as explored in Vermette et al. [59]). Future work should explore the feasibility and utility of finer-grained difficulty assessments by collecting suitably-labeled training data (e.g., advanced, intermediate, beginner tutorials) and using multi-class classifiers [16,22,52,64].

We have demonstrated that classifiers can automatically label a tutorial's difficulty; however, our approach does involve some manual work to initially build the classifier. In creating our command dictionary, we collected examples of "indirect" command reference by manually annotating a subset of tutorials (70 in total). While this command dictionary played a part in our best performing model (to calculate the Command Ratio feature), a model trained without it still achieved greater than 80% accuracy. Assigning human-readable labels to our LDA topics also involved a non-trivial amount of human labour. Future work could explore ways to automate this labeling to eliminate the need for expert inspection. One could also imagine using crowd workers [30] to assign topic labels.

Our study findings indicate that users appreciate having information on both a tutorial's difficulty level and its high-level topics. The combination of difficulty labels and topics has the potential to be particularly powerful in the context of feature-rich software given that a user's software expertise can vary substantially according to the topic [20]. A recommender system could also leverage such a classifier by using recent advances in expertise detection [18,19] and task detection [27,61] to automatically recommend tutorials.

IX. Limitations

Our investigation is largely limited to a single feature-rich application (i.e., Photoshop). While we applied our approach to a second application (Fusion360), the two applications share certain characteristics. Generalizability to different domains such as programming tutorials or "how-to" tutorials for more physical skills (e.g., such as knitting or "DIY" home projects) remains an area of future work. Future work should also verify the generalizability of our study findings to a larger sample size. Deploying *TutVis* would enable us to collect more ecologically valid data on how *TutVis* supports real-world tutorial browsing and selection. Our results indicate that our prototype helped increase users' confidence in their tutorial selection; however, this was based on short-term use where users did not have the opportunity to try out the tutorials. Future work should investigate how our approach to tutorial annotation impacts user confidence over the long-term as they use the tutorials to complete specific tasks and enhance their new software skills.

X. Conclusion

We presented an automatic, machine-learning approach to labeling an online software tutorial's difficulty. We showed our developed tutorial features could be leveraged to classify advanced vs. beginner Photoshop tutorials at 85% accuracy. Our system, *TutVis* represents only one point in the design space of how this expertise information might be used to support tutorial selection. With ongoing advances in software expertise detection, our approach paves the way for new technologies that match users with online resources that best suit their current levels of software expertise.

REFERENCES

[1] V. Balakrishnan and E. Lloyd-Yemoh. 2014. Stemming and lemmatization: A comparison of retrieval performances. In *Proceedings of SCEI Seoul Conferences*.

[2] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2012. Waken: reverse engineering usage information and interface structure from software videos. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 83–92.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3: 993–1022.

[4] Doris U. Bolliger and Supawan Supanakorn. 2011. Learning styles and student perceptions of the use of interactive online tutorials. *British Journal of Educational Technology* 42, 3: 470–481.

[5] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1: 5–32. https://doi.org/10.1023/A:1010933404324

[6] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18, 4: 467–479.

[7] Andrea Bunt, Patrick Dubois, Ben Lafreniere, Michael Terry, and David Cormack. 2014. TaggedComments: Promoting and Integrating User Comments in Online Application Tutorials. In *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI'14*, 4037–4046.

[8] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 93–102.

[9] Lynne Cooke. 2010. Assessing Concurrent Think-Aloud Protocol as a Usability Test Method: A Technical Communication Approach. *IEEE Transactions on Professional Communication* 53, 3: 202–215. https://doi.org/10.1109/TPC.2010.2052859

[10] Soussan Djamasbi, Marisa Siegel, and Tom Tullis. 2010. Generation Y, web design, and eye tracking. *International journal of human-computer studies* 68, 5: 307–323.

[11] Volodymyr Dziubak, Patrick Dubois, Andrea Bunt, and Michael Terry. 2016. Switter: Supporting Exploration of Software Learning Materials on Social Media. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, 1209–1220.

[12] Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2011. Searching for software learning resources using application context. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, 195.

[13] Adam E. M. Eltorai, Syed S. Naqvi, Soha Ghanian, Craig P. Eberson, Arnold-Peter C. Weiss, Christopher T. Born, and Alan H. Daniels. 2015. Readability of Invasive Procedure Consent Forms. *Clinical and Translational Science* 8, 6: 830–833. https://doi.org/10.1111/cts.12364

[14] Laura Faulkner and David Wick. 2005. Cross-user analysis: Benefits of skill level comparison in usability testing. *Interacting with Computers* 17, 6: 773–786. https://doi.org/10.1016/j.intcom.2005.04.004

[15] Adam Fourney, Ben Lafreniere, Richard Mann, and Michael Terry. 2012. Then click ok!: extracting references to interface elements in online documentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 35–38.

[16] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* 44, 8: 1761–1776. https://doi.org/10.1016/j.patcog.2011.01.017

[17] Arin Ghazarian and S. Majid Noorhosseini. 2010. Automatic detection of users' skill levels using high-frequency user interface events. *User Modeling and User-Adapted Interaction* 20, 2: 109–146.

[18] Jun Gong, Fraser Anderson, George Fitzmaurice, and Tovi Grossman. 2019. Instrumenting and Analyzing Fabrication Activities, Users, and Expertise. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*, 1–14.

[19] Tovi Grossman and George Fitzmaurice. 2015. An Investigation of Metrics for the In Situ Detection of Software Expertise. *Human–Computer Interaction* 30, 1: 64–102.

[20] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A Survey of Software Learnability: Metrics, Methodologies, and Guidelines. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, 649.

[21] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 143–152.

[22] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class AdaBoost. *Statistics and Its Interface* 2, 3: 349–360. https://doi.org/10.4310/SII.2009.v2.n3.a8

[23] Abram Hindle, Christian Bird, Thomas Zimmermann, and Nachiappan Nagappan. 2015. Do topics make sense to managers and developers? *Empirical Software Engineering* 20, 2: 479–515. https://doi.org/10.1007/s10664-014-9312-1

[24] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 256–265.

[25] Amy Hurst, Scott E. Hudson, and Jennifer Mankoff. 2007. Dynamic detection of novice vs. skilled use without a task model. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, 271. https://doi.org/10.1145/1240624.1240669

[26] Caitlin Kelleher and Randy Pausch. 2005. Stencils-based tutorials: design and evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 541–550.

[27] Md Adnan Alam Khan, Volodymyr Dziubak, and Andrea Bunt. 2015. Exploring personalized command recommendations based on information found in Web documentation. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, 225–235.

[28] Kimia Kiani, George Cui, Andrea Bunt, Joanna McGrenere, and Parmit K. Chilana. 2019. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*: 1–14. https://doi.org/10.1145/3290605.3300570

[29] Juho Kim, Philip J Guo, Carrie J Cai, Shang-Wen Daniel Li, Krzysztof Z Gajos, and Robert C Miller. 2014. Data-driven interaction techniques for improving navigation of educational videos. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, 563–572.

[30] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J Guo, Robert C Miller, and Krzysztof Z Gajos. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, 4017–4026.

[31] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012. Delta: a tool for representing and comparing workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1027–1036.

[32] Ben Lafreniere, Andrea Bunt, Matthew Lount, and Michael Terry. 2013. Understanding the Roles and Uses of Web Tutorials. In *Seventh International AAAI Conference on Weblogs and Social Media*.

[33] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community enhanced tutorials: improving tutorials with multiple demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1779–1788.

[34] Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic model. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, 1536–1545.

[35] Barry R. Lawson, Kenneth R. Baker, Stephen G. Powell, and Lynn Foster-Johnson. 2009. A comparison of spreadsheet users with different levels of experience. *Omega* 37, 3: 579–590.

[36] Wei Li, Tovi Grossman, and George Fitzmaurice. 2012. GamiCAD: a gamified tutorial system for first time autocad users. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, 103. https://doi.org/10.1145/2380116.2380131

[37] Wei Li, Justin Matejka, Tovi Grossman, Joseph A. Konstan, and George Fitzmaurice. 2011. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction* 18, 2: 1–35.

[38] EA Locke, GP Latham - American Psychologist, and Undefined 2002. 2002. Building a practically useful theory of goal setting and task motivation: A 35-year odyssey. *American psychologist* 57, 9: 705.

[39] Matthew Lount and Andrea Bunt. 2014. Characterizing Web-Based Tutorials: Exploring Quality, Community, and Showcasing Strategies. In *Proceedings of the 32nd ACM International Conference on The Design of Communication CD-ROM*, 6.

[40] Richard E. Mayer and Roxana Moreno. 2003. Nine Ways to Reduce Cognitive Load in Multimedia Learning. *Educational Psychologist* 38, 1: 43–52. https://doi.org/10.1207/S15326985EP3801_6

[41] J McGrenere and G Moore Interface. 2000. Are we all in the same" bloat"? *Graphics interface* 2000: 187--196.

[42] Qiaozhu Mei, Xuehua Shen, and Chengxiang Zhai. 2007. Automatic Labeling of Multinomial Topic Models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[43] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew Mccallum. 2011. Optimizing Semantic Coherence in Topic Models. In *Proceedings of the conference on empirical methods in natural language processing*, 262–272.

[44] Daša Munková, Michal Munk, and Martin Vozár. 2014. Influence of Stop-Words Removal on Sequence Patterns Identification within Comparable Corpora. In *International Conference on ICT Innovations*, 67–76. https://doi.org/10.1007/978-3-319-01466-1_6

[45] David Newman, Jey, Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic Evaluation of Topic Coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 100–108.

[46] David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. 2010. Evaluating topic models for digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries - JCDL '10*, 215.

[47] Jakob. Nielsen and Jakob. 1993. *Usability engineering*. AP Professional.

[48] Fred Paas, Alexander Renkl, and John Sweller. 2003. Cognitive Load Theory and Instructional Design: Recent Developments. *Educational Psychologist* 38, 1: 1–4.

[49] Amy Pavel, Floraine Berthouzoz, Björn Hartmann, and Maneesh Agrawala. 2013. Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials. In *Citeseer, Tech. Rep.*

[50] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F Cohen. 2011. Pause-and-play: automatically linking screencast video tutorials with applications. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 135–144.

[51] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, Rocco Oliveto, Mir Hasan, Barbara Russo, Sonia Haiduc, and Michele Lanza. 2016. Too long; didn't watch!: extracting relevant fragments from software development video tutorials. In *Proceedings of the 38th International Conference on Software Engineering*, 261–272.

[52] Anita Prinzie and Dirk Van den Poel. 2008. Random Forests for multiclass classification: Random MultiNomial Logit. *Expert Systems with Applications* 34, 3: 1721–1732. Retrieved October 6, 2019 from https://www.sciencedirect.com/science/article/pii/S0957417407000498

[53] Philipp Probst, Marvin N. Wright, and Anne□Laure Boulesteix. 2019. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9, 3. https://doi.org/10.1002/widm.1301

[54] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 248--256.

[55] Vidya Ramesh, Charlie Hsu, Maneesh Agrawala, and Björn Hartmann. 2011. ShowMeHow: translating user interface instructions between applications. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 127–134.

[56] Shahed Anzarus Sabab. 2019. An Investigation on Automatically Assessing an Application Tutorial's Difficulty. Retrieved May 14, 2020 from https://mspace.lib.umanitoba.ca/xmlui/handle/1993/34467

[57] Alexandra Schofield, Måns Magnusson, and David Mimno. 2017. Pulling Out the Stops: Rethinking Stopword Removal for Topic Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 432–436.

[58] Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management* 50, 1: 104–112. https://doi.org/10.1016/J.IPM.2013.08.006

[59] Laton Vermette, Shruti Dembla, April Y Wang, Joanna Mcgrenere, and Parmit K Chilana. 2017. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. In *Proceedings of the ACM : Human-Computer Interaction (1,1), Computer-Supported Cooperative Work and Social Computing (CSCW)*.

[60] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation Methods for Topic Models. In *Proceedings of the 26th annual international conference on machine learnin*, 1105--1112.

[61] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 285. https://doi.org/10.1145/3173574.3173859

[62] Sarah Weir, Juho Kim, Krzysztof Z Gajos, and Robert C Miller. 2015. Learnersourcing subgoal labels for how-to videos. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 405–416.

[63] Ryen W. White, Susan T. Dumais, and Jaime Teevan. 2009. Characterizing the influence of domain expertise on web search behavior. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining - WSDM '09*, 132.

[64] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability Estimates for Multi-class Classification by Pairwise Coupling. *Journal of Machine Learning Research* 5, Aug: 975–1005.

[65] sklearn.model_selection.StratifiedKFold — scikit-learn 0.21.3 documentation. Retrieved October 1, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

[66] Tobii Ghost - Stream with Eye Tracking. Retrieved September 11, 2019 from https://gaming.tobii.com/software/ghost/