박 사 학 위 논 문
Ph.D. Dissertation

인터랙션 디자인을 위한 순차적 지식 마이닝

Mining Sequential Knowledge for Interaction Design

2021

장 민 석 (張 珉 碩 Chang, Minsuk)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박사학위논문

인터랙션 디자인을 위한 순차적 지식 마이닝

2021

장민석

한국과학기술원

전산학부

인터랙션 디자인을 위한 순차적 지식 마이닝

장 민 석

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2020년 9월 9일

심사위원장	김 주 호	(인)
심 사 위 원	오 혜 연	(인)
심 사 위 원	차 미 영	(인)
심 사 위 원	Mira Dontcheva	(인)
심 사 위 원	Elena Glassman	(인)

Mining Sequential Knowledge for Interaction Design

Minsuk Chang

Advisor: Juho Kim

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Daejeon, Korea
January 7, 2021

Approved by

Juho Kim
Associate Professor, School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DCS
20145265

장민석. 인터랙션 디자인을 위한 순차적 지식 마이닝. 전산학부 . 2021년. 93+vi 쪽. 지도교수: 김주호. (영문 논문)
Minsuk Chang. Mining Sequential Knowledge for Interaction Design. School of Computing . 2021. 93+vi pages. Advisor: Juho Kim. (Text in English)

Abstract

From blogs to how-to videos to social media posts, it has become increasingly easy to share knowledge online. As a result, even for a single task seemingly as simple as baking chocolate chip cookies, we have access to hundreds of thousands to millions of tutorials in less than a second through a simple web search. Each tutorial shared online is not only a set of task instructions, but also a display of the authors' unique set of skills, constraints, and strategies. Consequently, the resulting diversity and scale of available online tutorials present users with both opportunities and challenges. Specifically, it remains difficult 1) to compare and analyze which tutorial to follow and invest the time to learn, and 2) to apply the picked tutorial to the current user's expertise, available tools, and other workflow contexts. However, the typical form factor for information retrieval systems like search engines and recommendation systems force users to go back and forth between the search results and individual tutorials to compare and analyze how they are different or similar. Also, users have a difficult time conceptualizing how different methods agree and disagree with each other and understand what makes one tutorial work and the other not work. However, whether people accomplish to find applicable tutorials depends on their ability to access not "the single right" tutorial, but on their ability to access and explore the "right set" of tutorials. This thesis presents systems that support users in forming hypotheses, comparing, and analyzing tutorials in different settings. I demonstrate techniques for extracting the nonlinear compositions of semantic substrates in large scale tutorials and demonstrations, such as task-specific domain knowledge, constraints, and users' strategies. Then I present abstract representations reconstructed with these semantic substrates and novel interface designs that strengthen people's ability to learn from tutorials.

Keywords Human-Computer Interaction, User-centered Data Structures, Sequential Knowledge Mining, Explorative Interaction, Interaction Design

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Online resources, such as tutorial videos and blog posts, are popular references people frequently visit during or before carrying out unfamiliar tasks. As the creation of tutorials are becoming easier with the advent of both general-purpose and domain-specific knowledge sharing platforms - as well as social media, the online tutorials exist in various communication modalities, in different contexts, with and without specific targeted audiences.

Tutorials not only detail sequential knowledge, but also is set of decisions. They have semantic structures characterized by the context of the tutorial that the user needs to be able to recognize. The different tool usages, expertise level, unique sets of skills, constraints and strategies employed by the tutorial creator that may be fully or partially observable in the tutorial content are important contextually dependent factors that contribute to how the sequential knowledge is expressed.

Yet, efficiently making informed decisions when interacting tutorials is still challenging. Difficulties arise partially because of the diversity and the scale of available online tutorials, and also because our current models of representing tutorials lack the ability to express the embedded diverse contextual information like strategies and constraints. Three common interaction scenarios are finding "the right" tutorial, navigating the tutorial, and applying the tutorial, all of which suffer from lack of systematic support. Specifically, it remains difficult 1) to compare and analyze which tutorial to follow and invest the time to learn, 2) to navigate the tutorial while learning the task at hand and following along, and 3) to contextualize the picked tutorial to the current user's expertise, available tools, and other workflow context. The goal of this dissertation is to develop principles and interactive tools that provide users with alternative interaction techniques for finding, navigating, and applying tutorials.

1.1 Challenges in Interacting with Tutorials

1.1.1 Comparing and Analyzing Tutorials

There are two common decision making situations when comparing and analyzing tutorials is important. First is when users search for which tutorial to follow and invest the time to learn, and second is when the at-scale analysis itself is the user task.

In today's interfaces, when users search for a tutorial regardless of the domain and format, the interface returns a list of results. Users have to examine the items one by one going back and forth between the search result and each of the time. As much as it sounds tedious, this is actually a very important activity. When users come back to the search result list after examining and item, they are forming hypothesis about which of remaining items are more likely to be the tutorial closest to what they need. But the information available for the users to form hypothesis are title, thumbnails, and maybe snippets of textual description.

Then the task itself is actually analysing large scale tutorials, like if it were an instructor or a TA deciding on which tutorials work best for the course, or your job is an analyst's job the interface support is as limited as the search scenario.

1.1.2 Navigating and Contextualizing Tutorials

Navigating a tutorial is not an easy task. Especially when users are trying to follow along the tutorial while carrying out the task, skimming and finding relevant information and jumping around are all difficult. This is because users need to switch frequently between carrying out the tasks and navigating the tutorials.

For example, when navigating and searching for a specific part of interest in a video tutorial, users have to rely on the timeline, either by sequentially scrubbing or click guessing, which are both quite inefficient and tedious – users often cannot exactly remember what's happening when in the tutorial especially if they're learners and it's their first time watching and, if they're following and doing the task at the same time. Also users can't search or navigate to parts they haven't watched or learned.

And users who seek for tutorials are more likely to be those who are not familiar with the task, it's difficult for them to internalize the tutorial and adapt to their own task scenario. Especially when users have more than one available tutorials at hand, because users can't follow more than one tutorials in parallel, it is difficult to contextualize and understand where the methods agree and disagree from each other, and understand what makes one tutorial work and the other not work for the user. When faced with problems like the tutorial step does not work or apply to the user's task, often the users do not know which of the previous actions of the task they've done is related to the problem they're faced with.

1.2 Thesis Contribution

This thesis makes a contribution by distilling users' information needs and the computational building blocks, and thereby solving the mismatch in three commonly observed tutorial interactions, namely finding, navigating, and applying tutorials.

To make comparing and analyzing usable, user support for understanding of both the landscape of available tutorials, similarities and differences between the tutorial items is needed. Specifically, helping users form hypotheses in-between browsing and conducting mini-experiments is crucial. To solve navigating and contextualizing, it requires user support for being able to efficiently formulate their own context to the system for matching those of the tutorials.

To computationally assist users' decision making in tutorial search, a "good representation" of the sequential knowledge that captures the above mentioned sources of variation in the tutorials is necessary. The unique approach used in this thesis uses **task specific knowledge unit**, which is a composite structure of elements that make up the tutorial but augmented with user's task semantics.

This dissertation contributes to the advancement of techniques of rich representations of tutorials for designing interactions for finding, navigating, and applying tutorials. The technical contributions are based on evidence collected through interviews with users in the domain and online surveys.

The dissertation makes the following technical contributions in three areas:

1. Techniques for analyzing tutorials at scale
   (a) Techniques for capturing domain knowledge using human computation and natural language processing
   (b) Techniques for calculating semantic and structural similarities between tutorials

(c) Visualizations techniques for analyzing tutorials at different granularities

2. Techniques for navigating tutorials using multimodal interfaces

   (a) Empirical taxonomy of navigational needs

   (b) Design recommendations for tutorial video navigation interface

   (c) Interaction techniques for in-video information seeking

3. Techniques for building user-centered knowledge graph

   (a) Techniques for capturing multistream user behavioral traces

   (b) Techniques for aggregating multi-user data

   (c) Novel interfaces for expanding users' workflow knowledge

This dissertation also provides evidence, through laboratory studies, that the introduced techniques are successful. In particular, the dissertation contributes:

- Evidence that the introduced tutorial representations and interactions are usable and expressive through laboratory evaluations with domain experts and learners.
- Evidence from a laboratory study that the techniques for in-video navigation alternativesn enable users to explore and efficiently find target information.
- Evidence from a laboratory studies that an interactive workflow knowledge leads to useful feedback for novices.

1.3 Thesis Overview

This section presents a brief overview of the structure of this dissertation by chapters.

Chapter 2 Background

Chapter 2 describes relevant research on systems and techniques for interacting with sequential knowledge and mining procedural content. Specifically, visual analytics for structural data, similarity comparison techniques for data structures, video browsing and navigation interfaces, designing voice user interface, software learning and workflow, also literature from learning at scale.

Chapter 3 RecipeScape

RecipeScape is an interactive system for browsing and analyzing the hundreds of recipes of a single dish available online. We also introduce a computational pipeline that extracts cooking processes from recipe text and calculates a procedural similarity between them. To evaluate how RecipeScape supports culinary analysis at scale, we conducted a user study with cooking professionals and culinary students with 500 recipes for two different dishes. Results show that RecipeScape clusters recipes into distinct approaches, and captures notable usage patterns of ingredients and cooking actions.

Chapter 4 Voice Navigation for How-to Videos

This chapter introduces three think-aloud studies using: 1) a traditional video interface, 2) a research probe providing a voice controlled video interface, and 3) a wizard-of-oz interface. From the studies, we distill seven navigation objectives and their underlying intents: pace control pause, content alignment pause, video control pause, reference jump, replay jump, skip jump, and peek jump. Our analysis found that users' navigation objectives and intents affect the choice of referent type and referencing approach in command utterances. Based on our findings, we present design recommendations such as 1) support conversational strategies like sequence expansions and command queues, 2) allow users to identify and refine their navigation objectives explicitly, and 3) support the seven interaction intents.

Chapter 5 RubySlippers

This chapter we introduce RubySlippers, a multimodal prototype that allows users to navigate how-to videos by mixing temporal referencing like "skip 30 seconds"and content-based referencing based on keyword algebra. RubySlippers allows users to use conversational description of the content they are looking for to navigate the how-to video. Users can also update their queries in a algebraic manner by adding and removing keywords. RubySlippers also suggests commands and keywords at each pause to inform the user about both the available command and the potential candidate target scenes. The computational pipeline uses keyword based similarity matching between every sentence in the autogenerated video caption and the users' navigation command utterance to infer the intended navigation targets. Our evaluation with 12 participants showed users are able to navigate quicker with lower mental demand with RubySlippers than existing time-based voice interfaces.

Chapter 6 Workflow Graphs

This chapter introduces *Workflow graphs*, or *W-graphs*, which encode how the approaches taken by multiple users performing a fixed 3D design task converge and diverge from one another. The graph's nodes represent equivalent intermediate task states across users, and directed edges represent how a user moved between these states, inferred from screen recording videos, command log data, and task content history. The result is a data structure that captures alternative methods for performing sub-tasks (e.g., modeling the legs of a chair) and alternative strategies of the overall task. As a case study, we describe and exemplify a computational pipeline for building *W-graphs* using screen recordings, command logs, and 3D model snapshots from an instrumented version of the Tinkercad 3D modeling application, and present graphs built for two sample tasks. We also illustrate how *W-graphs* can facilitate novel user interfaces with scenarios in workflow feedback, on-demand task guidance, and instructor dashboards.

Chapter 6 Discussions and Future Work

Chapter 6 discusses some of the insights that came out of building and testing the systems in this thesis. It also outlines avenues of future work on the systems and ideas in this thesis, in combination with the complementary work of others in this space.

Chapter 2. Background and Related Work

2.1 Interacting with Sequential Knowledge

2.1.1 Video Browsing and Navigation

Previous works have investigated interaction techniques for navigating videos beyond simple timeline interfaces. For example, Dragicevic et al. [28] found that direct manipulation of video content (via dragging interactions) is more suitable than direct manipulation of a timeline interface for visual content search tasks. Swift [71] and Swifter [72] improved scrubbing interfaces by presenting pre-cached thumbnails on the timeline. Together with the video timeline, video thumbnails [**?**] are commonly used to provide viewers with a condensed and straightforward preview of the video contents, facilitating the searching and browsing experiences. Hajri et al. [3] use personal browsing history visualizations to enable users to more effectively spot previously watched videos.

Similar to how our studies investigate voice UI patterns, Li et al. [64] investigated digital video browsing strategies using traditional mouse-based graphical user interfaces. They found the most frequently used features were time compression, pause removal, and navigation using shot boundaries. Crockford et al. [25] found that VCR-like control sets, consisting of low-level pause/play operations, both enhanced and limited users' browsing capabilities, and that users employ different playback speeds for different content.

A study on video browsing strategies reported that in-video object identification and video understanding tasks require different cognitive processes [27]. Object identification requires localized attention, whereas video understanding requires global attention.

2.1.2 Interaction Techniques for How-To Videos

Interactions with tutorials has been a popular research subject in the HCI community. Web tutorials serve a variety of needs from expanding skill sets to experiencing experts' practices [60].

MixT [20] automatically generates step-by-step mixed media tutorials from user demonstrations and Duploblock [44] infers and tracks the assembly process of a snap-together block model in real-time. Panopticon [45] displays multiple sub-sequences in parallel to present a rapid overview of the entire sequence.

For software tutorials, Nguyen et al. [78] found that users complete tasks more effectively by interacting with the software through direct manipulation of the tutorial video than with conventional video players. Pause-and-play [83] detected important events in the video and linked them with corresponding events in the target application for software tutorials. FollowUs [61] captured video demonstrations of users as they perform a tutorial so that subsequent users can use the original tutorial, or choose from a library of captured community demonstrations of each tutorial step. Similarly, Wang et al. [110] showed that at-scale analysis of community-generated videos and command logs can provide workflow recommendations and tutorials for complex software.

Specific to educational videos, LectureScape [53] utilized large scale user interaction traces to augment a conventional interface, while ToolScape [51] utilized storyboard summaries and an interactive timeline to enable learners to quickly scan, filter, and review multiple videos without having to play them.

2.1.3 Software Learning and Workflow Capture

Early HCI research recognized the challenges of learning software applications [14], and identified the benefits of minimalist and task-centric help-resources [13]. More recently, Grossman et al. [41] identified five common classes of challenges that users face when learning feature-rich software applications: understanding how to perform a task, awareness of tools and features, locating tools and features, understanding how to use specific tools, and transitioning to efficient behaviors.

Of the challenges listed above, the majority of existing work on assisting users to acquire alternative workflows has looked at how to promote the use of keyboard shortcuts and other expert interaction techniques [40, 67, 96, 66], with less attention on the adoption of more efficient workflows. Closer to the current work is CADament [65], a real-time multi-player game in which users compete to try and perform a 2D CAD task faster than one another. In the time between rounds of the game, the user is shown video of peers who are at a higher level of performance than they are, a feature which was found to prompt users to adopt more efficient methods. While CADament shares some similarity with the current work, the improvements were at the level of refining use of individual commands, rather than understanding alternative multi-command workflows.

Beyond systems explicitly designed to promote use of more efficient behaviors, a number of systems have been designed to capture workflows from users, which could then be made available to others. Photo Manipulation Tutorials by Demonstration [39] and MixT [21] enable users to perform a workflow, and automatically convert that demonstration into a tutorial that can be shared with other users. Meshflow [26] and Chronicle [42] continuously record the user as they work, capturing rich metadata and screen recordings, and then provide visualizations and interaction techniques for exploring that editing history. In contrast to these works, which capture individual demonstrations of a task, W-graphs captures demonstrations from multiple users, and then uses these to recommend alternate workflows. In this respect, the current work is somewhat similar to Community Enhanced Tutorials [**?**], which records video demonstrations of the actions performed on each step of an image-editing tutorial and provides these examples to subsequent users of the tutorial. However, W-graphs looks at a more general problem, where the task is not sub-divided into pre-defined steps, and users thus have much more freedom in how they complete the task.

Summarizing the above, there has been relatively little work on software learning systems that capture alternative workflows, and we are unaware of any work that has tried to do so by building a representation that encompasses many different means of performing a fixed 3D modeling task.

2.2 Analyzing Sequential Knowledge

2.2.1 Visual Analytics for Structural Data

Interfaces for analyzing instructions and workflows have been a subject of rich prior work. Sifter [80] is an interface for browsing, comparing, and analyzing a large collection of web-based image manipulation tutorials. Delta [57] is an interface for comparing pairwise similarities of image processing workflows. Visualizing histograms of parameters such as stroke length and brush sizes helped 3D artists to compare digital sculpting workflows [95]. Visualizing worker behavior and worker output in crowdsourcing workflow has been found to be effective for crowdsourcing quality control [90]. Visualizing sequences of intermediate steps students take in problem-solving helped identify different strategies and points of confusion [112]. Visual analytics tools can ana-

lyze temporal data, such as tracking and comparing different versions of a slide deck [30], mining statistical insight from event sequences [68], analyzing patterns in health records [97], and finding similar student records [31].

To support analysis of recipe instructions at scale, our approach extends existing research on interfaces for instruction analysis by combining visualization and analysis on clusters of instructions with the lower level analysis features.

2.2.2 Tree Representation and Similarity Comparison

There are two areas where tree representations are widely used to compare structural similarities: comparing phylogenetic trees in bioinformatics and detecting code clones in software engineering. Additionally, we discuss how tree edit distance could be applied to culinary analytics.

A phylogenetic tree is a branch diagram which represents evolutionary dependencies of biological species. Researchers in bioinformatics have been comparing phylogenetic trees by using statistical and structural metrics such as maximum-likelihood of evolutionary parameters [118], neighbor-joining [92], and nodal-distances [10].

Tree similarity comparisons are also popular in code clone detection. For judging structural similarities in code clones, algorithms use features like characteristic vectors [46], syntax patterns [59], and token sequences of syntax trees [7, 49]. For judging semantic similarities, algorithms use graph isomorphism methods to dependency graphs [89] of source code, where nodes represent expressions and statements.

To compute recipe structure similarity, we use a tree edit distance [102] method. Tree edit distance methods find a sequence of operations like addition, deletion, and substitution of nodes, each associated with a cost, which minimizes the total cost to convert one tree to another. Computing the edit distances between unordered, labeled trees is NP-Complete [122] even for binary trees with the label alphabet size of two. However, we build recipe representation as an ordered tree, i.e., the first child of every node is always a cooking action. We then employ polynomial-time algorithms for ordered labeled trees [121, 122] to calculate the edit distances.

2.2.3 Mining and Summarizing Procedural Content

A number of research projects have investigated how user-created procedural content can be analyzed or mined for useful information. RecipeScape [16] enables users to browse and analyze hundreds of cooking instructions for an individual dish by visually summarizing their structural patterns. Closer to our domain of interest, Delta [58] produces visual summaries of image editing workflows for Photoshop, and enables users to visually compare pairs of workflows. We take inspiration from the Delta system and this work's findings on how users compare workflows. That being said, our focus is on automatically building a data structure representing the many different ways that a task can be performed, rather than on how to best visualize or compare workflows.

Query-Feature Graphs [36] provide a mapping between high-level descriptions of user goals and the specific features of an interactive system relevant to achieving those goals, and are produced by combining a range of data sources, including search query logs, search engine results, and web page content. While this approach could be valuable for understanding the tasks performed in an application, and the commands related to those commands, query-feature graphs do not in themselves provide a means of discovering alternative or improved workflows.

Several research projects have investigated how to model a user's context as they work in a software application with the goal of aiding the retrieval and use of procedural learning content, for example using command log data [70], interactions gathered through accessibility APIs across multiple applications [37], or coordinated web

browser and application activities [34]. Along similar lines, Wang et al. [111] developed a set of recommender algorithms for software workflows, and demonstrated how they could be used to recommend community-generated videos for a 3D modeling tool. While the above works share our goal of providing users with relevant workflow information, their algorithms have focused on using the stream of actions being performed by the user, not the content that is being edited. Moreover, these techniques are not designed to capture the many different ways a fixed task can be performed, which limits their ability to recommend ways that a user can improve on the workflows they already use.

2.3 Domain Specific Knowledge

2.3.1 Learning at Scale

A final area of related work concerns how technology can enable learning at scale, for example by helping a scarce pool of experts to efficiently teach many learners, or by enabling learners to help one another. As a recent example, CodeOpticon [43] enables a single tutor to monitor and chat with many remote students working on programming exercises through a dashboard that shows each learner's code editor, and provides real-time text differences in visualizations and highlighting of compilation errors.

Most related to the current work are *learnersourcing* techniques, which harness the activities of learners to contribute to human computation workflows. This approach has been used to provide labeling of how-to videos [52], and to generate hints to learners by asking other learners to reflect on obstacles they have overcome [38]. The AXIS system [114] asks learners to provide explanations as they solve math problems, and uses machine learning to dynamically determine which explanations to present to future learners.

Along similar lines, Whitehill and Seltzer investigated the viability of crowdsourcing as a means of collecting video demonstrations of mathematical problem solving [113]. To analyze the diversity of problem-solving methods, the authors manually extracted the problem solving steps from 17 videos to created a graph of different solution paths. W-graphs produce a similar artifact for the domain of software workflows, and with an automated approach for constructing the graphs.

In summary, by capturing and representing the workflows employed by users with varying backgrounds and skill levels, we see W-graphs as a potentially valuable approach for scaling the learning and improvement of software workflows.

2.3.2 Data-Driven Culinary Analytics

The research community has investigated a wide range of computational methods for analyzing and mining cooking knowledge. For example, constructed using a large dataset of ingredients, ingredient networks [103] and flavor networks [1, 106] can judge which ingredients go well together and which ones do not. They can also be used to recommend recipes with replaceable ingredients [98]. Deep neural networks can be trained to translate recipes from one style of cuisine to another [50], or to generate flavorful and novel recipes as well as humans [82], or to generate a high-quality text recipe from a food image and vice versa [93]. Also, ingredient similarity can predict recipe ratings [120], and user reviews of recipes can predict their attributes [29]. Features extracted from the recipe text can predict the gender of the recipe uploader [87]. PlateMate [79] analyzes nutrition information from food photographs using crowdsourcing. PlateClick uses a quiz-like visual interface for comparing two food

images to elicit user's food preferences [117].

While most existing research focuses on analyzing recipes using ingredient similarities and food image similarities, RecipeScape introduces *structural similarity* to data-driven culinary analytics. We use a tree (Figure 3.2) to represent a recipe structure. In RecipeScape, the structural similarity of recipes is dependent on both syntactic similarities from the tree shape, and on semantic similarities between the node labels that represent ingredients and actions.

2.3.3 Designing Voice User Interfaces

Most recent work on voice interfaces is done in an "assistant" context. For example, Porcheron et al. [85] studied how voice-based virtual assistants are made accountable to and embedded into conversational settings such as dinner tables. Myers et al. [76] also reported that while natural language processing errors occur the most, other types of errors frustrate users more, and users often take a guessing approach when voice interfaces fail. Moreover, nearly one quarter of all user-assistant exchanges were initiated from implicit conversational cues rather than from plain questions [107]. Users frequently use a diverse set of imprecise temporal expressions in both communication and planning, and have a variety of expectations about time inputs for virtual assistants [88].

Researchers have also successfully implemented voice user interfaces for specific user facing tasks. PixelTone [62] enabled users to edit photos using both speech and direct manipulation. ImageSpirit [19] enabled users to verbally refine image search results using the automatically extracted labels. Apparition [63] enabled users to sketch their interface, describe verbally where crowd workers and sketch recognition algorithms translate the input into user interface elements, add animations, and provide Wizard-of-Oz functionality.

Most voice user interfaces adapted a conversational agent like form factor like Amazon's Alexa, Apple's Siri or Google Assistant. While previous works have shown the effectiveness of voice interaction in assisting user tasks such as image editing [62] and parsing images [**?**], most interactions for video and audio control are primarily for basic content playback controls [8].

Commonly reported user problems in using voice interactions are discoverability of available commands [23], balancing the tradeoff between expressiveness and efficiency [73]. As a remedy displaying available voice commands when you touch the tools [100], and vocal shortcuts [56] have been shown to be effective. However, these methods only work when the tasks are software tasks making them difficult to apply for physical tasks how-to videos.

For general voice user interface, common recovery strategies are hyperarticulation and rephrasing [76], both of which usually does not lead to a different outcome. Although the experiment was done with chatbots, and not with voice assistants, providing options and explanations as means of repairing a broken conversation were generally favored by users [5].

There has been many guideline level suggestions for how to design voice interactions. For example, guiding users to learn what verbal commands execute VUI actions and learn the actions supported to accomplish desired tasks with the system are important [77]. Also, allowing users to recognize and recover from errors is just as important as preventing user errors, and flexibility and efficiency of use is needed [75].

One of the most relevant building blocks of our approach is the analysis of navigation behavior using voice. Instead of watching these how-to videos passively, viewers actively control the video to pause, replay, and skip forward or backwards while following along the video instructions. Based on these interaction needs, previous work has proposed usage of conversational interfaces for navigation of How-To videos for physical tasks [18].

Voice interface for navigation how-to videos remains underexplored, and no concrete designs have been introduced yet. We provide interpretations of the design recommendations specifically for navigating how-to videos with voice interface, and demonstrate how they can be realized with a prototype implementation.

Chapter 3. RecipeScape

3.1 Introduction

Cooking recipes provide ingredients and step by step instructions for making a dish, and thousands of recipes are available even for a single dish on the Internet. For example, searching for chocolate chip cookie recipes on Yummly[1] yields 40,000 recipes that span different sets of ingredients, required skills and tools, levels of detail, and even varying arrangements of commonly used cooking actions and ingredients for the dish. These recipes are naturally crowdsourced instructions for a shared goal. Their variety and scale present an opportunity to understand usage patterns of cooking actions and ingredients for different approaches to cooking a dish.

Imagine a chef who wants to be creative with chocolate chip cookies to develop a new dessert menu. The chef has many options to consider, for example, baking unique looking cookies, making a pie using cookies as the crust, or using a specific type of dough that doesn't require baking. Where should the chef start to research the different ways to make or make use of chocolate chip cookies? Imagine a culinary student who is asked to cook a classic tomato pasta and an exotic tomato pasta for an assignment. What is the difference between the set of recipes titled "classic" versus those titled "exotic"? While thousands of recipes for a single dish are available online, it's difficult to browse, compare, and analyze them for coming up with new ideas or interpreting different cooking processes and their results.

For cooking professionals and culinary students, discovering usage patterns of cooking actions and ingredients to understand their implications is just as important as preparing a delicious meal. From our interviews with 10 cooking professionals, we learned that to mine and understand diverse cooking processes, they compare and analyze recipes in three different levels of granularity][pi]; *groups of recipes, individual recipes, and individual cooking actions or ingredients.* From a professional chef's menu research activities to training in culinary schools, a wide range of cooking practices emphasize reinterpretation of dishes. Common approaches include applying unusual cooking actions to usual ingredients, applying usual cooking actions to unusual ingredients, or both. These tasks require grouping recipes into similar operational patterns of cooking actions and ingredients, in-depth investigation of individual recipes, and browsing and comparison of individual cooking actions or ingredients.

In this paper, we present RecipeScape (Figure 3.1), an interactive tool for analyzing hundreds of recipes for a single dish. RecipeScape provides three main visualizations, addressing each of the three data granularity levels in recipe analysis; RecipeMap (Figure 3.1a) presents a bird's-eye view of *recipes in clusters* generated by the system. Each point on the map is a recipe, and the distance between them indicates their similarity. RecipeDeck (Figure 3.1b) enables an in-depth inspection and pairwise comparison of *individual recipes*. RecipeStat (Figure 3.1c) visualizes usage patterns of *individual cooking actions and ingredients*. Providing these visualizations requires processing and analyzing a large number of recipes. To achieve this goal, we present a computational pipeline (Figure 3.6) that scrapes recipes available online, converts them into a tree representation, and computes pairwise similarities. The pipeline represents each recipe as a tree (Figure 3.2) to capture the structural information (e.g., a sequence of actions, a set of ingredients involved in an action) embedded in a recipe. For accurate tree construction, we employ a machine-crowd workflow to label cooking actions and ingredients in recipe texts using

[1] http://www.yummly.com

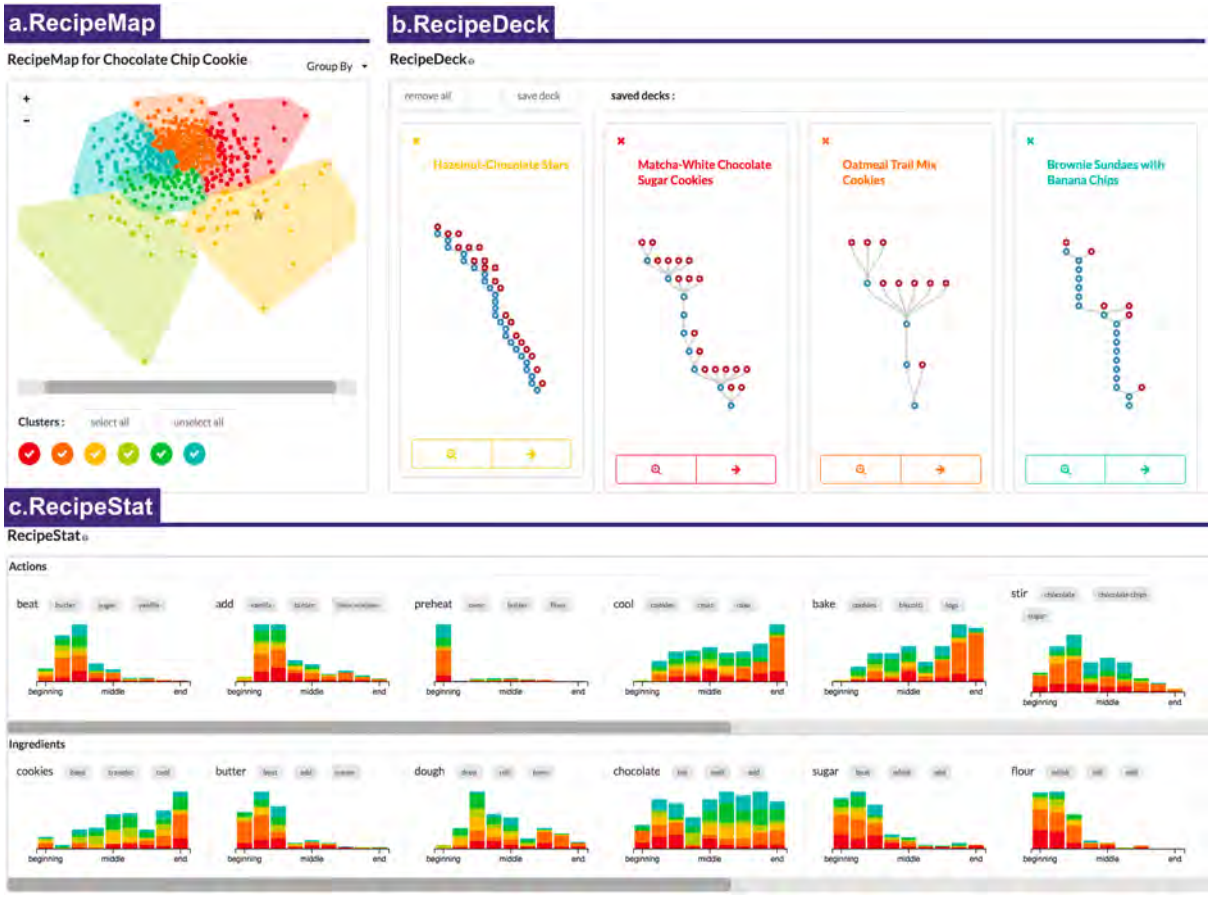


Figure 3.1: RecipeScape is an interface for analyzing cooking processes at scale with three main visualization components: (a) RecipeMap provides clusters of recipes with respect to their structural similarities, (b) RecipeDeck provides in-depth view and pairwise comparisons of recipes, and (c) RecipeStat provides usage patterns of cooking actions and ingredients.

a custom annotation interface(Figure 3.7).

In our crawled dataset of 487 chocolate chip cookie recipes and 510 tomato pasta recipes, 27,879 verbs are tagged by the Stanford CoreNLP's Part-of-Speech tagger [69]. Among them, our pipeline identified 14,988 verbs as relevant cooking actions. Also, our pipeline corrected 9,987 cooking actions that were mislabeled by the tagger, which is 40% of the cooking actions in the dataset.

In a qualitative evaluation with cooking professionals and culinary students, we asked participants to carry out a series of browsing and comparison tasks as well as to freely explore for new discoveries. Participants found data-driven evidence for subjective attributes of recipes like "general recipe" or "exotic recipe". They also discovered usage patterns of cooking actions and ingredients that distinguish one recipe cluster from another, by combining insights from the three visualizations of RecipeScape.

This paper makes the following contributions:

- **Design goals** for systems that aim to support analysis for cooking professionals by examining recipes in aggregate. These are identified from interviews with professional chefs, patissiers, cooking journalists, recipe website managers, and food business researchers.

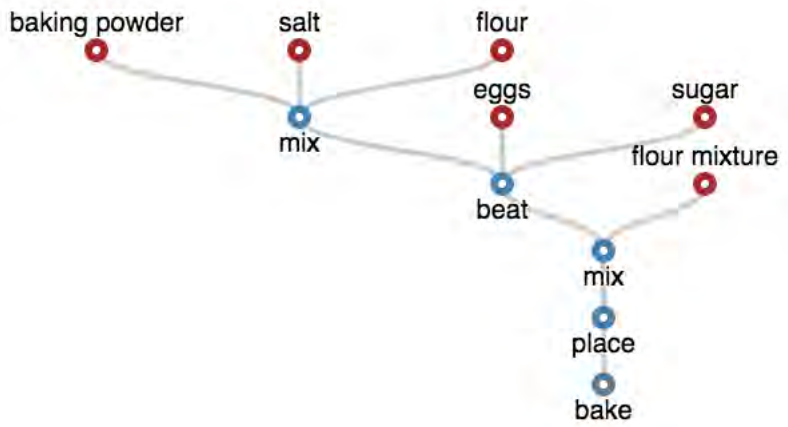


Figure 3.2: A tree generated by RecipeScape for "Daniela's Brownie" recipe, https://www.epicurious.com/recipes/food/views/danielas-brownies-104362.

- **RecipeScape**, **an interactive visual analytics interface** that enables browsing, comparison, and visualization of recipes at scale and supports analysis on three different levels of data granularity: *clusters of recipes, individual recipes, and individual cooking actions and ingredients.*
- **A computational pipeline** that scrapes recipe instructions from online websites, extracts the cooking action and ingredient information using a machine-crowd workflow, translates them into a tree representation, and computes similarities between pairs of recipes.

3.2 Formative Study

We conducted a series of interviews with cooking professionals to understand how they use online recipes in their current analysis practices and how they might benefit from using them more efficiently.

3.2.1 Interviews

We interviewed 2 professional chefs (10 or more years of experience), 1 patissier (5 years of experience), 2 cooking journalists (20 years of experience, 5 years of experience), 2 recipe website managers (3 years of experience), and 3 food business researchers (15 or more years of experience).

Each session took approximately 90 minutes and included a semi-structured interview followed by a feedback session on the general idea of exploring recipes at scale.

We asked our participants (1) what kinds of recipe analysis are involved in their daily job and what their current practices are, (2) what additional analysis would make their job more convenient, and (3) what analysis could increase and expand their capabilities.

After open-ended discussions, we presented component sketches designed to support recipe exploration at scale to encourage further discussion and ground their feedback. We used these low-fidelity sketches since exploring recipes at scale is an unaccustomed concept even for professionals. The participants were encouraged to think

aloud while they were browsing and comparing recipes using the components presented in the sketches[2].

3.2.2 Results

Current Practice: Casual but Interrogative Browsing

When developing new menus, professional chefs and patissiers said they compare recipes and search for unusual ingredients or creative uses of usual ingredients. A professional chef noted, *"There's no such thing as a completely novel recipe"*, and emphasized the importance of casually browsing a variety of recipes to maximize exposure to diverse recipes even for a single ingredient or a single dish.

When planning an article about a specific dish, both cooking journalists explained that they browse more than ten different recipes to understand common cooking actions, common ingredients, and common tips and hacks. They visit restaurants and ask the chef about the recipe, or use cookbooks and publicized recipes by famous chefs to grasp these characteristics. When they write articles with specific themes, they sometimes ask chefs to either devise or introduce recipes that are unfamiliar to the general public.

Recipe website managers examine recipes and label them with tags for feeding the recipes into their search engine. This is manually done, because existing algorithms available to end users cannot accurately provide rich and appropriate tag labels.

In summary, cooking professionals with analytical needs commonly search for unusual ingredients and cooking process, casually browsing a variety of recipes for a single dish.

Desired Information: Statistics of Recipes

While all participants agreed they want to be able to easily find recipes with uncommon cooking actions and ingredients, their reasons are different. We found professional chefs and patissiers rely on reverse engineering a dish to study unusual usages of a specific cooking skill or an ingredient. But reverse engineering requires a lot of trial and error and exact replication is very hard to achieve. Cooking journalists are interested in creative variants or unusual reinterpretations of a dish to be able to introduce them to the public. However,"creative" and "unusual" are very subjective measures, and they normally spend weeks trying to find something unimaginable for their audience. Recipe website managers are interested in standard recipes, and evidence to claim "standardness" of the recipes.

Food business researchers want to measure to what extent a de-facto standard version of a dish has been established. They explained if recipes are more standardized for a specific food, it is likely to be a saturated market, whereas if the recipes vary, the corresponding food business is still in its growth stage. It is used as one of the metrics to evaluate the market cycle and to predict an upcoming trend in the food business.

All participants want a categorization feature. Suggested ideas of categorization criteria include specific ingredient constraints like "gluten-free" or "sugar-free", cooking tool constraints like "no oven" or "microwave only", and types of cuisine. They also want more subjective criteria like uniqueness, difficulty of the recipe, and different ways of cooking the dish.

In summary, recurring needs expressed by the professionals are methods for discovering common and uncommon cooking actions and ingredients, and clues from which they could answer questions like how a set of recipes differ from another set of recipes, and what factors contribute to the difference. Also, participants want to

[2]The component sketches are available in the supplemental materials.

easily discover ”average” or ”standard” versions of a dish, and evidence of subjective attributes like uniqueness or difficulty of recipes.

3.2.3 System Design Goals

Two researchers iteratively analyzed the interview data more than four times in total with an interval of at least two days between sessions to enhance validity. We identified 52 topics during this process, and subsequently clustered these topics into 13 different themes.[3] From the 13 topics, we focused on the needs that would benefit most from exploring recipes at scale. We excluded several classes of topics beyond the scope of this work: (1) those requiring information that is unavailable in online recipes, such as trends and cultural information; (2) those relevant to real-time cooking support like hacks and mistakes; and (3) those already supported by existing systems, like categorization criteria and ingredient-based information.

Four researchers independently brainstormed hypotheses and frames of explanation for the remaining topics. Through rounds of discussion, we agreed that the data granularity framework is most explanatory. Then we classified the remaining topics into their data granularity and derived the three design goals.

The interview results emphasize the need to provide users with an interactive analytics system that enables browsing and comparing recipes. Based on the analysis of the interviews and the participants' suggestions, we identified three design goals for **tools to support recipe analysis at scale**. The individual design goals address three different levels of data granularity for recipe analysis: ingredients and cooking actions (D1), recipes (D2), and clusters of recipes (D3).

D1. **Provide statistical information about ingredients and cooking actions** to support analysis at the individual ingredient/cooking action level, such as answering questions like “what are some unusual ingredients?” and “what are some unusual cooking actions?”.

D2. **Provide in-depth examination and comparison of individual recipes** to support instruction level analysis, such as answering questions like “what are the detailed step by step instructions of this recipe?” and “what are shared instructions and ingredients between two recipes?”.

D3. **Provide analysis for recipes in aggregate** to support cluster level analysis and between-cluster similarities and across-cluster differences, such as answering questions like “what makes a set of recipes standard of the dish?”, “what are some creative variants of this recipe?”, and “what are different ways of cooking the dish?”

3.3 RecipeScape

To address these design goals, we present RecipeScape (Figure 3.1), an interactive tool that enables browsing, comparison, and analysis of recipes at scale. RecipeScape provides three main components: RecipeMap, RecipeDeck, and RecipeStat. RecipeMap (Figure 3.3) provides an overview of recipes with cluster information. RecipeDeck (Figure 3.4) provides information about individual recipes with the original description, the corresponding tree representation, and pairwise comparison of recipes. RecipeStat (Figure 3.5) provides the usage patterns of cooking actions and ingredients in the selected clusters of recipes from RecipeMap.

[3]The 52 topics and 13 themes are available in the supplemental materials.

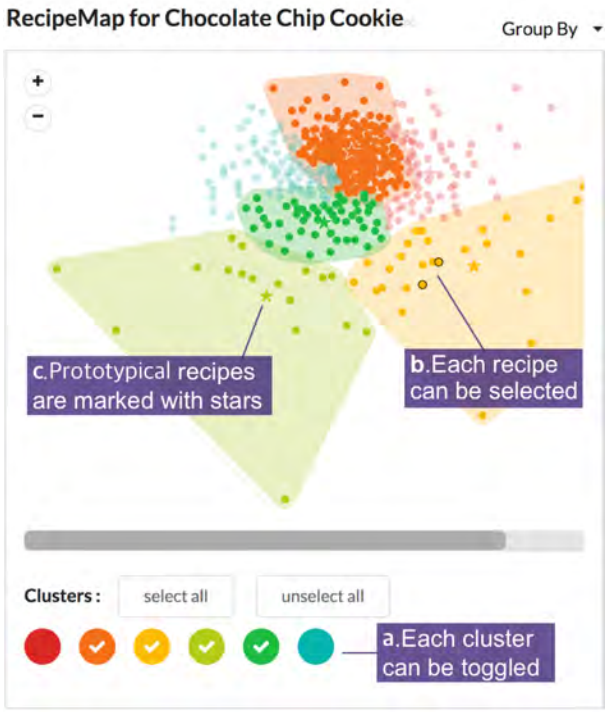


Figure 3.3: RecipeMap provides clusters of recipes with respect to their structural similarities.

3.3.1 RecipeMap

RecipeMap (Figure 3.3) supports queries related to D3 (Provide analysis for recipes in aggregate), e.g., the prototypical recipes, outlier recipes, and a bird's-eye view of different clusters of recipes for a specific dish. Each point on the map is a recipe. The distance between recipes reflects the structural similarity of the respective recipe instructions. The prototypical recipe, i.e., most structurally representative one in each cluster, is marked with a star (Figure 3.3c). Users can select clusters by clicking on the color key at the bottom (Figure 3.3a). Users can also select any recipe on the map (Figure 3.3b) for an in-depth view. When one or more clusters are selected, RecipeStat updates to only reflect the information in the chosen clusters. Similarity metrics and clustering algorithms are discussed in the Computational Pipeline Section.

3.3.2 RecipeDeck

RecipeDeck (Figure 3.4) supports queries related to D2 (Provide in-depth examination and comparison of individual recipes). User-selected recipes in RecipeMap are added here (Figure 3.4b), with a default view of the tree representation. The tree in this view does not have any labels, allowing users to focus on structural comparisons of multiple recipes on RecipeDeck. Users can click on the magnifier icon (Figure 3.4c) for a detailed popup with the recipe text and the labeled tree representation. Users can also click on the right arrow icon to view textual instructions without invoking the popup. Furthermore, users can perform a pairwise comparison of two recipes by selecting two recipes on RecipeDeck, and then clicking "compare". Upon clicking, a popup (Figure 3.4a) opens with a side-by-side comparison of cooking action sequences and an ingredient comparison of the two recipes in a Venn diagram.

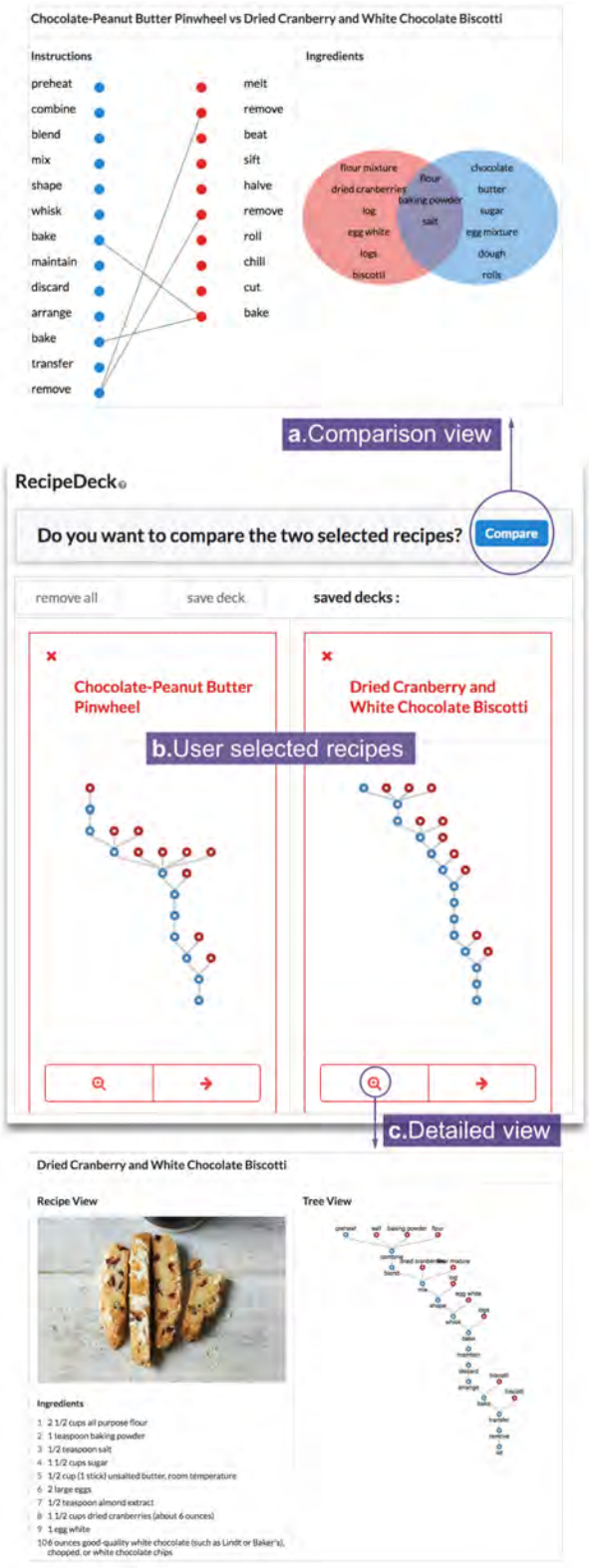


Figure 3.4: RecipeDeck: RecipeDeck displays (b) user selected recipes and provides (c) a detailed view and (a) pairwise comparisons of recipes.

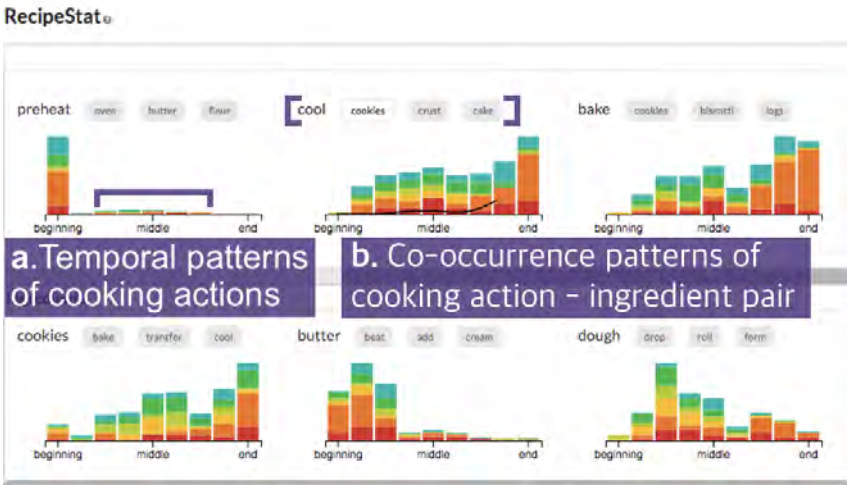


Figure 3.5: RecipeStat provides temporal usage trends of cooking actions and ingredients, and co-occurrences patterns of cooking action-ingredient pairs.

3.3.3 RecipeStat

RecipeStat (Figure 3.5) supports queries related to D1 (Provide statistical information about ingredients and cooking actions). For the 10 most used cooking actions in the selected clusters of recipes, users can examine how each cooking action is used at different stages of the cooking process in recipes of the selected cluster. This information supports the discovery of different approaches to cooking a dish. For example, users can click on the recipes with "Preheat" occurring in different stages of the cooking process (Figure 3.5a). Users are able to hover the bars on the histogram to see which recipes correspond to the specific selection. The corresponding recipes are highlighted in RecipeMap, and are added to RecipeDeck with a click. Users can also click on the top three most used ingredients next to each cooking action label to view the usage pattern of cooking action-ingredient pairs like cool-cookies (Figure 3.5b). When the cluster selection changes, RecipeStat recalculates the statistics and redraws the histogram. The same visualization and functionality is also provided for the 10 most used cooking ingredients.

3.4 Computational Pipeline

In this section, we describe the underlying pipeline (Figure 3.6) of RecipeScape for constructing graphical representations of recipes and obtaining similarity metrics by highlighting the data gathering, parsing, and similarity comparison steps.

3.4.1 Data Gathering

In the data gathering step, we crawl all search results for a queried dish, like chocolate chip cookie and tomato pasta, from recipe websites that use the schema.org's Recipe scheme[4]. Schema.org's schemes are agreed templates for storing structured data, with specific document elements like ingredients and instructions. Most major recipe websites use the Recipe scheme as their internal representation, which makes using it for data gathering step more generalizable.

[4]http://schema.org/Recipe/

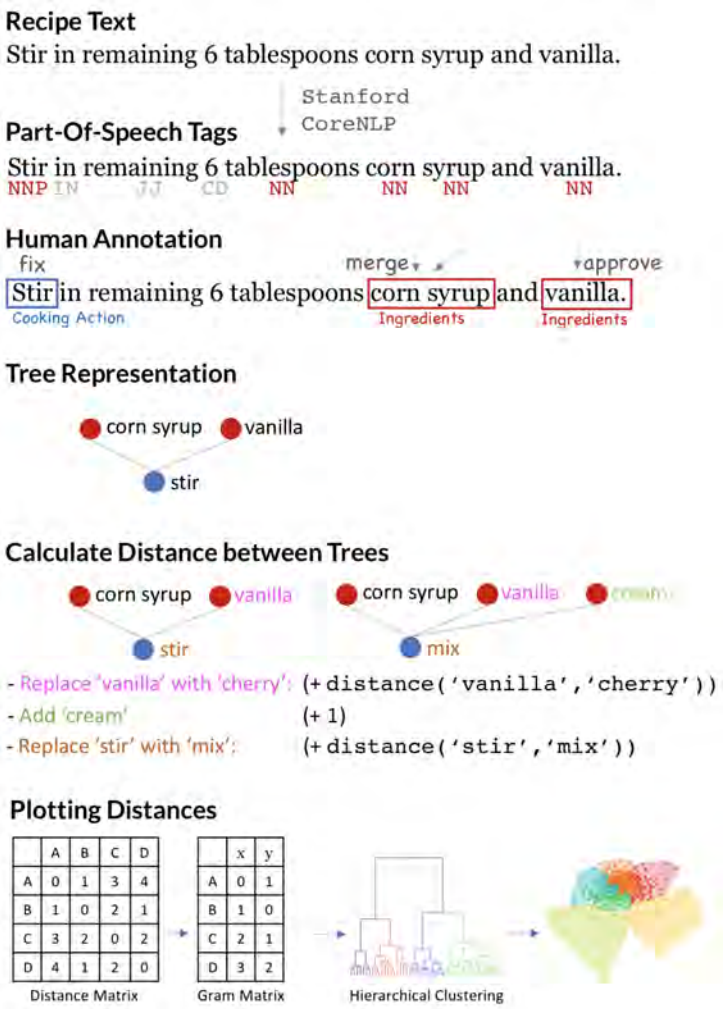


Figure 3.6: Our computational pipeline combines Part-of-Speech tagging and human annotation to convert recipe text into a tree representation, and calculates pairwise distance between recipes.

3.4.2 Parsing

In the parsing step, we use a Stanford CoreNLP [69]'s Part-of-Speech (POS) tagger to label verbs and nouns in recipe instructions crawled in the data gathering step. Most state-of-the-art POS taggers are statistically trained using mostly declarative sentences. As a result, their performance is rather limited with imperative sentences in recipe instructions like "Whisk in chocolate hazelnut spread until combined and remove from heat.". The parsers recognize "Whisk" as a noun and "spread" as a verb, but they are a verb and a noun in this sentence, respectively. To overcome this drawback and to more accurately identify tokens for cooking actions and cooking ingredients, we recruited 12 annotators to use a custom web-based interface (Figure 3.7) for annotating recipe instructions. After an iteration with the POS tagger, the crowd annotator fixes the tags that are labelled incorrectly. The parser then generates an ordered tree representation for each recipe, where the first child of every node is a cooking action and the siblings are the ingredients involved in that action.

There are several reasons we decided to use a tree structure over a sequence. Our preliminary study [17] using a sequence of cooking actions and ingredients, and string edit distances did not yield meaningful clusters

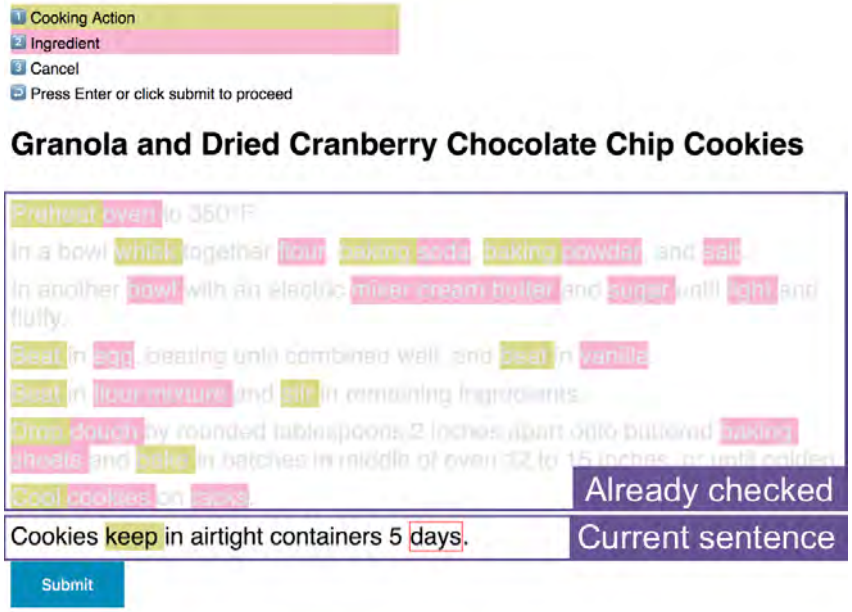


Figure 3.7: A web-based annotation interface for correctly labeling recipe actions and ingredients.

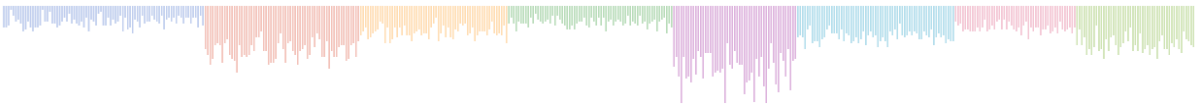
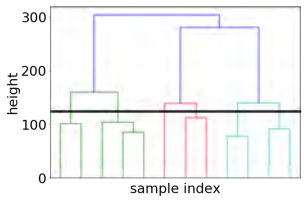


Figure 3.8: Lengths of recipes and their cluster membership shown with colors: this clustering result is based on sequence representation, and are dominated by the sequence lengths.

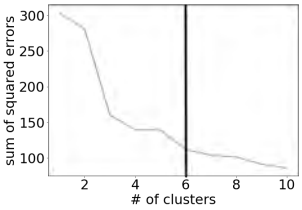
to users, meaning the sequence representation did not capture meaningful structural differences in the cooking context. The clustering result was highly sensitive to the length of the sequences (Figure 3.8), which motivated us to consider a branching structure rather than a linear structure. Also, there exists information that cooking actions or ingredient alone cannot capture. For example, generic action verbs like "place", "remove", and "heat" yields different interpretations depending on whether they are associated with ingredients or cooking tools. Thus it is advantageous to incorporate a hierarchical structure.

3.4.3 Similarity Comparison

In order to obtain similarities between recipes, we use a tree edit distance [121], a commonly used technique for comparing tree structures. It finds a sequence of operations like adding, removing, and relabeling nodes, each associated with a cost, which minimizes the total cost to convert one tree to another. To incorporate the semantic difference between individual cooking actions and ingredients in capturing the structural difference, we dynamically adjust the weights associated with the relabel operations. These weights are calculated from a pre-trained word embedding model with one million recipe instructions [93]. We use the cosine similarity between two words in the embedding space as the weight associated with relabel operations; the associated cost then is $(1 - weight)$, because weight is the similarity, and the cost is the difference. This is motivated by intuition that a resulting structure from replacing "mix" with "add" should be considered more similar than that of replacing



(a) A Dendrogram



(b) The Elbow Method

Figure 3.9: (a) Dendrogram for chocolate chip cookie recipes. (b) Elbow Method: a plot of unexplained variance vs number of clusters: adding another cluster at six clusters does not improve the validity. From both (a) and (b), six clusters seem reasonable.

"mix" with "heat". For add and remove operations, we assign a unit length cost of 1. This discourages adding and removing of nodes and promotes relabeling of nodes. It is another attempt at making difference measures less sensitive to lengths of the structure, a limitation we encountered when using a sequential representation.

This similarity information is stored in a pairwise distance matrix, where each element is the tree edit distance between the corresponding recipes. The distance matrix is then converted into x,y coordinates using the Gram matrix [105, 24] to preserve distance information. We use the calculated coordinates to plot the recipes on RecipeMap for a bird's-eye view of structural similarities between all recipes. To highlight the structurally different clusters of recipes, we use hierarchical clustering [48]. Hierarchical clustering methods do not require the predetermined number of clusters before the analysis, which is common in other popular clustering methods, e.g., K-means clustering. With hierarchical clustering, the researcher can choose the most appropriate number of clusters that suits their analytical purpose after calculating the similarities in the data. We found this post-analysis control over the number of clusters advantageous in this study, because we do not know how many distinct approaches to cooking a dish exists in advance. To decide the number of clusters to display to users, we used both dendrogram [81] and the elbow method [104]. A dendrogram is an arrangement of the clusters produced by hierarchical clustering based on a distance metric. The elbow method provides a graph of the amount of variance explained by the number of clusters. There is no definitive answers to how many clusters should be selected, because the interpretation of the resulting hierarchical structure is context-dependent and often several solutions are equally good. For our chocolate chip cookie example, the dendrogram (Figure 3.9(a)) suggests six clusters are a reasonable choice. Consulting the elbow method plot (Figure 3.9(b)), adding another cluster at six causes a minimal change on the slope, so we pick six clusters.

3.4.4 Pipeline Results

Dataset: We crawled 487 recipes for "chocolate chip cookie" and 510 recipes for "tomato pasta" from epicurious.com. We chose chocolate chip cookie and tomato pasta because they are popular and accessible dishes.

Parsing Accuracy: The ground truth label for all 214,109 tokens in the crawled cooking recipes are unavailable to assess the precision/recall accuracy. However, we made a significant improvement over the baseline parser. Out of 27,879 verbs tagged by the Stanford CoreNLP's POS tagger [69], 14,988 were cooking actions confirmed by human annotators. This means only 54% of the machine-tagged labels are relevant to culinary analysis. Furthermore, human annotators corrected 9,987 cooking actions the NLP tagger mislabeled. This counts up to

40% of the final 24,975 cooking action verbs used in the study, which represent improvements realized by human annotation.

3.5 RecipeScape Project Website

We provide links to our dataset, source code repository, and dashboard interface at https://recipescape.kixlab.org/.

3.6 Evaluation

RecipeScape is a tool for open-ended discovery by exploring cooking recipes of a single dish at scale and is not designed as an assistant to improve cooking skills. Hence, we investigated the effectiveness of novel exploration techniques for the professional analytic needs with open-ended qualitative studies rather than a task-based evaluation that measures an improvement over a baseline. Goals of evaluation were (1) to assess the feasibility of representing cooking process as a tree, and (2) to gain feedback on the effectiveness of RecipeScape in answering the following analytical questions that follow from the three design goals:

Q1. What are patterns of common and unusual ingredient and cooking action usage?

Q2. What are different ways of cooking a dish?

Q3. What are representative recipes of cooking a dish?

Q4. What are outlier recipes of cooking a dish?

Q5. What are the simplest and most complex recipes of cooking a dish?

Q6. What is the evidence for answers in Q1-Q5?

3.6.1 User Study with Cooking Professionals

We reached out to the same experts we interviewed in the earlier stage of the research for understanding analysis tasks of cooking professionals. Among them, two recipe website managers, one professional chef, and one cooking journalist participated in the interface evaluation study followed by a semi-structured interview, which lasted two hours. Experts were given a 5-minute tutorial of the interface and asked to freely explore and evaluate RecipeScape. They were asked to choose one or more clusters on RecipeMap and find characteristics that define the cluster. Experts were encouraged to think aloud as they browsed and compared recipes, and how they interpreted the findings.

3.6.2 Lab Study with Culinary Students

We invited 7 culinary students in a 90-minute session each. Participants were first asked to fill out a questionnaire on their current practices of searching and browsing recipes, i.e., how they search for recipes, when and how often they search for recipes. They were given a brief tutorial of the RecipeScape interface. Then they were asked to explore and use the interface. After participants indicated that they felt confident using the interface, we gave them 30 minutes to evaluate the interface by carrying out tasks to answer the questions Q1 to Q6 outlined

above. A session ended with an interview to understand deeper the observed interface usage patterns, and solicit qualitative feedback about the interface.

3.6.3 Results

We summarize the results and present main findings with respect to the three design goals, patterns of tool usage, and usability and usefulness of RecipeScape.

D1. Ingredient and action level analysis

The professional chef made an observation that recipes with must-have ingredients are plotted at the center of RecipeMap, and the outer ring of the recipes have additional ingredients that go well with the dish but are not necessary. The chef found the recipes on the edges to have ingredients that reflect more personal preferences, such as use of goat cheese and artichoke for pasta. The chef was also surprised to see recipes that use salt in the later stages of cooking pasta in RecipeStat, as opposed to the convention of using it in earlier stages, e.g. cooking pasta noodles in salted water. He mentioned, *"This is a professional tip that good restaurants use to make the first spoon of pasta taste extra sweet. If you put salt on tomato, it really brings out the sweetness. I'm surprised this hack is captured."* For one cluster of chocolate chip cookie recipes, the cooking journalist wanted to find cookies with decorations, and examined RecipeStat for recipes where "cover" was mostly used in later stages of the recipes. The corresponding highlighted recipes in RecipeMap agreed with her hypothesis. One student participant used the identical approach to find recipes with sugar frosting.

D2. Individual recipe level analysis

The professional chef spent significant time examining individual recipes near the edge of RecipeMap. He noted *"We (professional chefs) sometimes start from a specific main ingredient and seek creative interpretations. I find these recipes near the edges are more exotic."*

The cooking journalist frequently used the pairwise comparison of two adjacent recipes on RecipeMap to examine replaceable ingredients and actions.

Four out of seven student participants found the tree diagram in the in-depth view especially helpful for grasping the overall process of individual recipes and they felt confident about cooking the dish only by looking at the tree. One student specifically noted, *"I find recipes in the usual text format hard to visualize the process, because the ingredient sections and the instruction sections are separate. But this tree diagram summarizes the process very well, I can easily picture the cooking process."*

D3. Cluster level analysis

Two experts and five out of seven student participants mentioned the clusters reflect different ways of cooking very well. After selecting a cluster from RecipeMap, experts examined RecipeStat and formed hypotheses of what some characterizing attributes of the cluster might be. Then they used other components to verify their hypotheses. For example, the cooking journalist looked at one cluster and noticed there are baking soda and baking powder in the most used ingredients list in RecipeStat. She immediately mentioned, *"Recipes in this cluster probably do not use any eggs and probably involve baking in the later stages."*, which was confirmed by examining the recipes in the cluster in detail. The journalist also found a cluster where water was in the most used ingredients list in

RecipeStat. She then checked whether there is "chill" or "cool" in RecipeStat for cooking actions. When she found "chill", she mentioned *"These are the recipes for more crispy cookies. You use water so the ingredients don't stick as much, resulting in crispy cookies. This kind of dough tastes better when you cool them."* After reviewing a few recipes in the cluster, the hypothesis was verified. The professional chef discovered "salted water" in RecipeStat for one cluster. He then mentioned *"I would trust the recipes in this cluster more than the other ones. The fact that people described salted water, not just water, implies the instructions are more friendly and detailed."* Upon examining a few recipes in the cluster, the recipes were indeed more detailed than the others.

Patterns of Tool Usage

Every participant started from RecipeMap by choosing cluster(s) of their interest. Then they would select recipes in the center of the clusters and examine them in detail. Some would repeat these steps back and forth, but RecipeStat was always used in the last stage. When asked, participants explained RecipeMap is a good place to start analysis due to its similarity presentation. We learned the design of RecipeStat assumed knowledge of histograms, which some participants did not have. Participants explained they needed to examine a few individual recipes in depth to understand the information displayed in RecipeStat, which led to the usage pattern.

Usability and Usefulness of RecipeScape

The professional chef and cooking journalist noted RecipeScape would be useful for learning about recipes. The professional chef mentioned he would use this tool to understand a dish that he does not have much experience in. The cooking journalist wanted to use RecipeScape for Bibimbap, a traditional Korean dish. She explained there are many recipes of Bibimbap in English, because it is internationally popularized. She said RecipeScape would reveal diverse approaches that capture how this traditional dish is interpreted outside Korea.

Six student participants noted they want to use this tool in their studies if it supported dishes of their interest. They found similar recipes being located closely together in RecipeMap to be useful in comparison to existing services they use, because browsing in RecipeScape does not involve going back and forth between the list of search results and the specific recipe page. Three student participants also mentioned RecipeScape would be useful when they're preparing for cooking contests, where they have to reinterpret an existing dish. Using RecipeMap to explore various recipes helped them not only brainstorm ideas but also simulate how their interpretations can be translated into recipes. For example, one participant said *"I had thought about using liquor for making creative cookies (as an assignment), but was not sure how I could do it. I was able to spot a recipe that uses liquor (Frozen Grand Marnier Torte with Dark Chocolate Crust and Spiced Cranberries) just by browsing the recipes outside (near the edges), and it helped me understand better how liquor could be used in cookies."*

3.7 Discussion

We discuss findings, generalizability and possible limitations of this work.

3.7.1 Expert Knowledge in Naturally Crowdsourced Data

We observed an interesting example of an expert knowledge recovered in the analysis of crowd generated recipes supported by RecipeScape. Butter cream, made by beating the butter, is added in the last step of making

Figure 3.10: Usage patterns of two contrasting action-ingredient pairs (beat-butter and drop-dough) in chocolate chip cookie recipes.

the dough, because it inhibits gluten formation. According to RecipeStat, the beat-butter pair (Figure 3.10a) occurs mostly in the beginning of the cooking process, and drop-dough (Figure 3.10b), an intermediate product of making the dough, occurs after the creamed butter is made. Extending from this observation, it would be meaningful to further identify and characterize expert knowledge that is transferred and not transferred for informing further interaction designs around instructions.

3.7.2 Structural Representation and Parser Accuracy

In this research, the low accuracy of the NLP parser in generating the structural representation could make the analysis challenging. Ill-structured text and non-standard phrasing can lead to mislabels. To minimize confusion in such cases, we show the original recipe text next to the tree diagram in the detailed view for the parser errors to have less impact on user tasks. With a better performing Part-of-Speech tagger that successfully detects action verbs in imperative sentences, RecipeScape can immediately benefit from the algorithm by replacing the parser module.

3.7.3 Explainability of Clustering Algorithm

RecipeScape only provides clusters of recipes and does not provide explanations of the clustering results. In the end, users will have to make sense of the clusters generated by the algorithm. While this is inevitable for all clustering algorithms, we do have control over choosing the number of clusters after the similarity calculations. A number of ways can improve the explainability. Supporting manual labelling, or accompanying carefully designed topic modelling to generate themes for each clusters, or both could benefit the users.

3.7.4 Generalizability of the Pipeline

While RecipeScape is focused on culinary analytics, our pipeline is generalizable and could potentially apply to analyzing other instructions at scale. Researchers have explored how to present other kinds of procedural instructions like image manipulation tutorials, sculpting workflows. It varies in degree, but even in assembly or in photo manipulation tutorials where it seems like there is only one correct sequence of operations, there are often multiple feasible solutions. For example, when assembling a chair, one can start from the legs, the back, the armrests. A systematic analysis of instructions across domains is open to future study, but we believe our approach is still applicable to other domains.

3.7.5 Scalability of the Pipeline: More Data Dimensions

Our structural similarity comparison of tree representations allows adding more dimensions like time or tools. There remains a design decision of whether to treat these dimensions as separate nodes or as parameters of cooking action. Regardless, dynamically retrieving edit distance weights from a vector embedding space handles the semantic similarities of different dimensions. However, visualizing multiple dimensions and presenting them with meaningful interaction is an open challenge, which we hope to address in future work.

3.8 Conclusion and Future Work

This paper presents RecipeScape, an interactive system for analyzing hundreds of recipes for a single dish by visualizing summaries of their structural patterns. Our user study with cooking professionals and culinary students demonstrates that RecipeScape provides data-driven evidence to usual and unusual ingredients and cooking actions, common and exotic recipes, and different approaches to cooking a dish.

Chapter 4. Voice Navigation for How-to Videos

4.1 Introduction

People are increasingly turning to online how-to videos as guides for learning new skills or accomplishing unfamiliar tasks. YouTube searches for how-to videos are growing 70% year to year and as of 2015, 67% of the millennials believe they can find anything they want to learn on YouTube [74].

Instead of watching these how-to videos passively, viewers actively control the video to pause, replay, and skip forward or backwards while following along with the video instructions. We identified that these active control moments arise when 1) the viewer is presented with non-tutorial content (e.g., chit-chat and general introductions), 2) the viewer fails to match the pace, content, or context of the video (e.g., needing more time to complete a step, not understanding the instruction, or needing to compare the outcome in the video with the viewer's own result), and 3) the viewer is only interested in a specific part of the tutorial (e.g. a particular method or step). Using traditional mouse and keyboard interfaces, viewers commonly navigate tutorial videos by either sequentially scrubbing through the video timeline to examine the preview thumbnails or click-guessing on the timeline until the resulting video frame matches the desired point of interest.

However, many how-to videos teach *physical* tasks (e.g., playing an instrument, applying makeup, etc.) that involve interaction with real world objects. When following along with such videos, viewers need their hands both to execute the task and to control the video. Since they can't do both at once, viewers must alternate between the two operations. In having to alternate between these two operations, viewers incur a costly context switch: they must stop concentrating on the task itself to instead concentrate on controlling the video. Controlling the video alone can be both difficult and tedious using traditional timeline-based interfaces. This work specifically focuses on exploring navigation solutions for how-to videos for *physical* tasks.

Voice-based user interfaces (e.g., Apple Siri, Amazon Alexa, and Google Assistant) are becoming increasingly ubiquitous in commercial devices and provide a potential alternative for controlling how-to videos of physical tasks. Current voice-based video navigation systems (e.g., those in the web accessibility initiative [108], or virtual assistants with displays like Amazon Echo Show) support basic operations for video navigation such as pause, play, rewind, or fast forward (e.g., by 20 seconds). While these systems provide some help in the context of how-to videos, they are not specifically designed for this domain. Rather, they directly translate of low-level remote control operations (pause, play, etc.) into voice commands. An important question is whether this low-level remote-control-like interface is suitable for voice-driven video navigation interfaces for how-to videos, and if not, how should a useful voice interface for navigating how-to videos be designed?

In this work, we explore the mental models users have when navigating how-to videos, and report how the navigation objectives affect the command utterances. In particular, we answer the following research questions:

RQ1 What are the different types of navigation objectives and user intentions when "actively controlling" how-to videos?

RQ2 Do these objectives and intents affect users' linguistic choices for voice commands? If so, what linguistic characteristics are related to the objectives and intents?

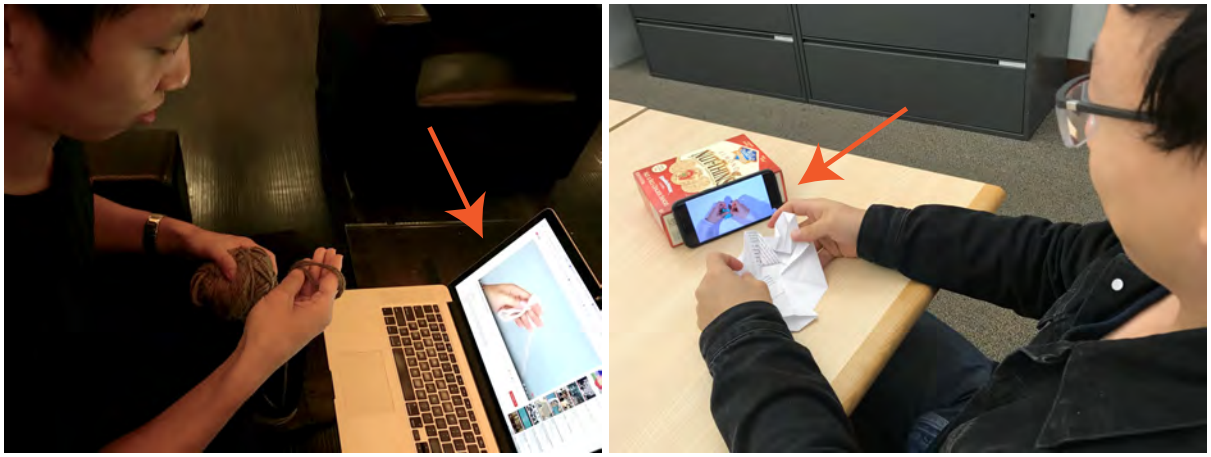


Figure 4.1: People use how-to videos to accomplish a variety of physical tasks. The person in the left photo is using the video on his laptop to learn how to stitch while the person on the right is attempting to fold an origami turtle by following the video on his phone. Since both of these are hands-on tasks, it would be difficult for these people to navigate the videos using traditional click or touch based input modalities.

RQ3 What are the challenges and opportunities for designing voice navigation interactions for how-to videos?

To this end, we report findings from three user studies. We conducted a think-aloud study of 20 participants using a YouTube interface to accomplish a how-to task, a 16-participant think-aloud study using a custom-built voice based video navigation research probe, and a Wizard-of-Oz study with 6 participants. From these studies, we distill a list of design recommendations for how to build a navigation system for how-to videos with voice and visual feedback.

In summary, our contributions include:

- A range of user objectives and contexts for navigating how-to videos. They are pace control, content alignment, video control, reference, replay, skip, and peek.
- An analysis of how these objectives and contexts, when applied to voice interactions for how-to video navigation, affect linguistic characteristics of user command utterances.
- A set of design recommendations for voice interaction with how-to videos.

4.2 Study 1 - Understanding How People Currently Navigate How-To Videos

In the first study, we asked participants to perform a physical task by watching a how-to video with a conventional mouse interface and the standard YouTube video player. Our primary observation points were when and how users pause and jump during the session.

In this experiment, we focused on two specific how-to tasks: learning to play a song on an instrument and learning to apply makeup. We recruited 20 participants on usertesting.com, an online user study platform. We recorded participant screens and think-aloud audio for all sessions. The music experiment consisted of 10 participants (8 male, 2 female, average age: 40, min: 21, max: 71) who regularly watch tutorial videos to learn

Task (# of participants)	Music (10)	Makeup (10)
Total minutes of tutorial	46:47	78:34
Total # of pauses	33	65
Total # of jumps	27	16
Per minute pauses	0.42	1.38
Per minute jumps	0.47	0.21

Table 4.1: Frequency of user pauses and jumps in a traditional interface

how to play songs on their musical instruments. The makeup experiment consisted of 10 participants (all female, average age: 33, min: 21, max: 56) who regularly watch makeup tutorial videos.

We instructed participants to select a video of their choice from YouTube that consisted of an unfamiliar song or an unfamiliar makeup style, respectively. They then had to follow along with the video tutorial and describe their thought process out loud. We specifically asked participants to explain what they were trying to achieve whenever they controlled the video (i.e., pause, play, rewind, etc.).

4.2.1 Findings

Participants picked tutorial videos with average lengths of 4 minutes 40 seconds (0:29 min, 12:08 max) for music tutorials, and 7 minutes 51 seconds (3:39 min, 11:04 max) for makeup tutorials. The average session length was 15 minutes 32 seconds (10:15 min, 25:55 max) for learning a song, and 20 minutes 21 seconds (10:42 min, 40:58 max) for learning a makeup look. Participants spent on average 3 times the length of the video following it.

General Impressions

For the task of learning a song, we observed that participants tended to either try to get a rough understanding of the entire song or to focus on learning a specific part. Two participants pointed out that this is due to an inherent characteristic of the task where it would take days or even weeks of practice for them to fully learn an entire song.

We also found that content of the video as well as task characteristics affected the distribution of interactions. For example, as seen in Table 4.1, the music session users jumped backward or forward over twice as often per minute as those in the makeup task. We observed this is because for makeup how-to videos, the distinction of where each step begins and ends is much more apparent as each step builds on top of the previous step. This allows natural pauses in between steps, giving space for users to catch up. In fact, per minute of viewing, users paused three times as often in the makeup task as the music task. Music how-to videos usually do not contain explicit steps, making the beginning and end points of a navigation unit ambiguous and more user dependent. Also each makeup how-to video was specific to certain aspects of makeup in general, e.g., eye makeup or contour makeup, whereas music tutorials usually try to tackle the entire song in one video.

Types of Pause Interactions

In this experiment, we observed 98 total pauses across both tasks (Table 4.1). From these tasks, we observed three different types of pause interaction.

Pace Control Pause. The most common type of pause was a pause to gain more time (78 of 98 pauses: 29 of 33 in music, 49 of 65 in makeup). This happens when the user understands the video content but fails to match

the pace of the video. With this pause, the user is trying to finish one step before moving onto the next. Unlike other types of pauses, in a pace control pause, the user is usually detached from the video, while concentrating on the physical task. Once users are caught up to the video using pace control pauses, they often end the pause by pressing play and without performing any other type of video navigation.

Content Alignment Pause. The second type of pause is a pause to compare what's in the video with what's in the hands of the user. This pause precedes the user checking to make sure that their state is similar to that of the video. For example, after the pause, users say *"I'm just trying to see if this is what he—the video instructor—has done."* or *"I need to see if this is it."* while making the comparison between what's in the video and what's in the hands of the user. Users often observe the paused video frame several times during these pauses. Content alignment pauses made up 9 out of 98 total pauses observed: 2 of 33 in music, 7 of 65 in makeup. Out of the 9 pauses of this type, 7 pauses were at the end of a step, right after the next step has begun, where the information in the still frame has not made a full transition yet. During the content alignment pauses, the user attention is split between the video and the physical task.

Video Control Pause. The final type of pause we observed is a pause for further video control. Reference jumps In this case, the user pauses the video and searches for the next navigation target point on the timeline by either guess-clicking, or scrubbing and examining the thumbnails. In this use case, the user's attention is entirely in the video. Video control pauses occurred in 8 of 98 total pauses observed: 1 of 33 in music, 7 of 65 in makeup. Video control pauses are always followed by a jump interaction described in detail in the next section.

Types of Jumping Interactions

In this experiment, we observed 43 total jump interactions from both tasks (Table 4.1). These jumps are broadly split into forward and backward jumps, and we break down the different user motivations that we observed. Users carried out *jumps* by pressing right or left arrow keys on the keyboard, or by clicking on the point of interest on the timeline, or by scrubbing the timeline.

Reference Jump. The first type of jump we observed is a *reference jump*. We observed 5 (out of 43) reference jumps (3 in music, 2 in makeup). In this case, the user jumps backwards in the video to remind themselves of something they saw in the past. Users typically only need to see a still image of the video for this jump.Usually a forward jump back to the original position is followed by a reference jump to continue where they left off.

Replay Jump. A *replay jump* is a different form of backward jump, where the user wants to re-watch a segment of the video again. We observed 24 (out of 43) replay jumps (21 in music, 3 in makeup). This jump happens when the user needs to get a better understanding, clarify a possible mistake, or to assure that the current understanding is correct. This jump if often followed by a play or a pause interaction.

Skip Jump. A *skip jump* is a type of forward jump where the user wants to skip content that is less interesting, like the introduction of the channel or the personal life of the YouTuber. We observed 10 (out of 43) replay jumps (2 in music, 8 in makeup). When the goal is to skip introductory content, the target is almost always "the beginning of the actual tutorial". Since the user cannot tell where exactly "the actual tutorial" begins, skip jumps happen in multiples. This forward jump often is followed by another skip jump or a play interaction.

Peek Jump. The second type of forward jump is a *peek jump*, where the user wants to skip ahead to see what the user should expect after performing one or a number of steps. We observed 4 (out of 43) replay jumps (1 in music, 3 in makeup). This happens when users want to check the intermediate or the final result in order to prepare and also check if the user is on the right track. The goal is not to skip the current step, but rather to help

Main Command	Popular Variants
play	resume, go, start, begin
pause	stop, wait, hold on
mute	volume off
unmute	volume on
louder	volume up
quieter	volume down
fast forward	skip ahead, skip
rewind	go back, back
faster	speed up
slower	slow down

Table 4.2: List of commands supported by our system

by anticipating future steps. A peek jump is often followed by a jump back to the original position in the video.

Other Interactions

Users sometimes paused the video to get the surprise introduction of an additional tool or material like a guitar capo or an unconventional makeup tool. We observed this 3 times (1 of 33 in music, 2 of 65 in makeup). Users also sometimes let the video play while concentrating on the physical task without paying much attention, but still listening to it. We observed this 26 times (10 in music, 16 in makeup).

4.3 Study 2 - Understanding how People Navigate How-to videos Using a Basic Voice UI

Results of our first study show that people often stop and jump within the videos, which requires frequent context switches. To understand what differences might be observed in users' thoughts and preferences of *voice* interactions in navigating how-to videos, we built a voice-enabled video player as a research probe. This research probe served as a "tools for design and understanding" [109] not a prototype interface to suggest new interaction techniques. We used our research probe as an apparatus to observe and elicit similarities and differences in user behavior in relation to the different types of pauses and jumps observed with a traditional mouse-based interface.

With our research probe, the user can play, pause, jump backward and forward by specifying the jump interval in seconds or minutes, speed up, slow down, mute, unmute, increase volume, and decrease volume. We used a grammar consisting of popular variations of the above commands (Table 4.2). We piloted the research probe and enumerated the list of command variants, and iterated until no new variant was observed. The interface also indicates when it is listening 4.2 and transcribes commands in real time to provide visual feedback to the user. Transcription uses HTML5 Web Speech API. Figure 4.2 shows the research probe interface.

We conducted a study that mirrored the previous one, except that participants were asked to use our voice-based research probe instead of YouTube, and a third task, knitting, was added to cover a more diverse set of tasks. We recruited 16 participants in total. The music task consisted of 7 (4 male and 3 female, average age of 35) participants, the makeup task consisted of 4 (all female, average age of 26) participants, and the knitting

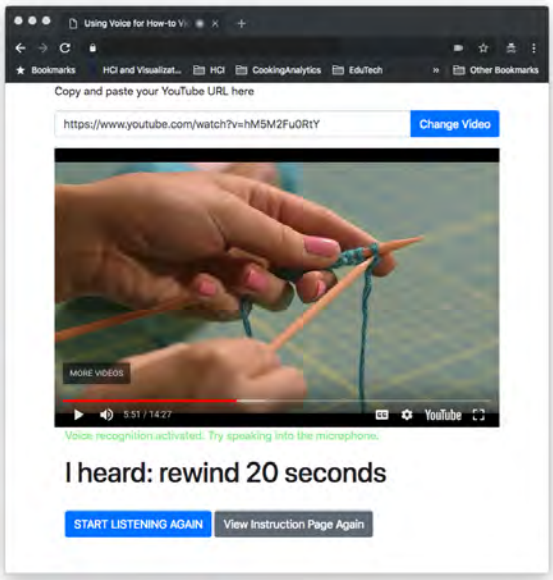


Figure 4.2: Our research probe allows users to select a YouTube video of their choice and control it with voice. The interface also indicates when it is listening and transcribes commands in real time to provide visual feedback to the user. For example, "I heard: rewind 20 seconds".

task consisted of 5 (all female, average age of 31) participants. None of the participants had participated in our previous experiment.

To minimize priming participants to use specific utterance choices, they were not instructed on what voice commands were available when communicating with the interface. When the system did not support a specific command, participants were instructed to think aloud and carry on.

Similar to the first study, we annotated each interaction and their occurrence counts, focusing on the three types of pauses and four types of jumps we have identified. Repeated utterances due to speech recognition failure were counted as only once. To further break down the composition of each command utterances, we annotated how users made references to navigation targets when they performed jump interactions using the features presented in Table 4.3.

4.3.1 Findings

The average length of the tutorial videos participants picked was 9 minutes 2 seconds (4:56 min, 15:29 max) for music tutorials, 13 minutes 47 seconds (10:09 min, 17:45 max) for the makeup tutorials, and 14 minutes 44 seconds (6:21 min, 23:10 max) for the knitting tutorials. The average session length was 25 minutes 39 seconds (17:29 min, 41:23 max) for learning a song, 34 minutes 54 seconds (26:24 min, 43:44 max) for learning a makeup look, and 26 minutes 5 seconds (16:22 min, 33:05 max) for learning a new knitting pattern.

Types of Pause Interactions

We observed that the command "stop" is used mostly for **video control pauses** (18 out of 25 "stop"s) where the command was followed by a jump. In contrast, "stop video" was used mostly for **content alignment pauses**, where the command was followed by a play command (13 out of 15 "stop video"s).

Features	Tags
Referent Type	Time, action, object
Referencing Styles	Specify an interval, specify a point, contextual description, content description, no referent
Reference Direction	Backward (rewind), forward (fast forward)
Causes	Mismatch in pace, mismatch in content, mismatch on context

Table 4.3: Features used for analyzing referencing utterances. Since we are interested in how users make references to what and why, user utterances are annotated with one tag from each of the features.

We also observed that users use the word "stop" to indicate a higher sense of urgency, or a need to navigate to a very specific point in the video. Here are some of the example use cases we've observed:

1. "Go back by a little", "Go back by a bit", "Stop"
2. "Stop", "I want to stop at this specific note (piano)"
3. "Stop", "I'm missing something here"
4. "Stop", "I don't know what's going on"

Participant 3 in the makeup experiment is an illustrative example. This participant used "stop video" and "play video" ten times each throughout the session to follow along the instruction. But when she needed to quickly go back and check how mascara was done in a hurry before moving on to the next step, she used "stop!" followed by "I need to go back to mascara now, I'm doing something different".

We found that both "pause" and "pause video" were frequently used for **content alignment pauses** and **pace control pauses**, "pause" was used 24 times out of 43, while "pause video" was used 10 times out of 12 for these jumps.

Types of Jump Interactions

Two frequently used commands for backward jumps were "go back" and "rewind". In this experiment, we observed 23 **replay jumps** and 28 **reference jumps**. We noticed for replay jumps, users use less concrete commands than for reference jumps, such as, "start from beginning", "let me see that again". "go back **about** 30 seconds", "go back **just a bit**", "go back **by little**", "go back **to the beginning**".

However, for **reference jumps**, users tend to be more specific, and repeat multiple times to find the exact target, using commands like "go back 30 seconds" and "go to 2 minute mark". Users also repeat concrete backward commands to find a specific desired position. Also, some users said "go back to where I paused" to go back to the original position in the video before the backward jump, which indicates the user is expecting the system to remember this position when performing jumps.

We observed forward jumps that refer both to contextual details, as well as concrete units of time. Examples are "skip about 30 seconds", "skip to the lesson". "skip to the makeup (as opposed to cleansing)", and "fast

Task (# of participants)	Music (7)	Makeup (3)	Knitting (6)
Total minutes of tutorial	63:25	41:21	88:28
Total # of pauses	46	32	28
Total # of jumps	43	22	21
Per minute pauses	0.73	0.78	0.31
Per minute jumps	0.68	0.54	0.24

Table 4.4: Frequency of user pauses and jumps in voice-enabled interface

forward a bit", and "skip to next step." We could not observe any different linguistic pattern between **skip jumps** and **peek jumps**.

One reason for this might be because users do not know the target position of the peek or the skip because they are both in the future (later in the video). In contrast, backward jump targets are usually those users have already seen once, which enables users to refer to their memory for more specific descriptions.

General Impressions

Participants found the concept of using voice to navigate how-to videos useful for their tasks. From the music experiment, P3 noted *"it's an interesting experience having to stop and play video without taking my hands off my guitar, it's wonderful."* and P4 also noted *"'this is a very powerful tool, especially if you're doing something with your hands."* From the makeup experiment, P3 reported *"I really like that I can get my products ready without touching the video"*. From the knitting experiment, P1 commented *"I love being able to use voice to control the video while I'm knitting so I don't have to stop from knitting."*.

We also noticed users would "stop" or "pause" the video before jumps a lot more often while using voice user interfaces. Jumps with specific references like "go back 20 seconds" is dependent on both the current position and the target, and without the pause the current position would keep changing, resulting inconveniences to adjust the interval or make multiple subsequent jumps. With the mouse interactions, in contrast, users are only specifying the target position and not the origin.

4.4 Study 3 - Understanding Expectations of Voice UI for How-to Videos

From the previous study, we learned that users' navigation intents affect their linguistic choices for command utterances. We also observed that commonly supported voice commands are limited to simple words, that it can be difficult for users to express their intents with a restrictive command space, and that it is difficult for systems to understand the intents. For example, different backward jump intents for "stop" and "pause" can only be understood in context of other commands before and after the stop, specifically analyzing preceding and succeeding commands and user goals, which is impractical in application settings where users need systems to understand the user intents in real time.

To inform how to disambiguate voice commands and corresponding user intents for navigating how-to videos, we conducted a Wizard-of-Oz experiment to learn how users would naturally converse for video navigation in the absence of these constraints. Participants were instructed to find a knitting video appropriate to their expertise

Challenges	Opportunities
Problems from interacting with video	Visual feedback strategies
Problems from interacting with voice	Conversational strategies
Problems from interacting with wizard	Wizard strategies

Table 4.5: Resulting code book for analysis of Wizard of Oz Study(Study 3)

level, and follow the video while performing the task.

We invited 6 participants (3 male, 3 female), 5 of whom were complete novices in knitting, and 1 of whom was a hobbyist with self-reported expertise level of intermediate. A researcher was sitting next to the participant as the wizard video controller, watching the tutorial video with the participant. The participant could only control the video by talking to the wizard video controller. Users were encouraged to converse without any technical barriers in mind. We also conducted semi-structured interviews at the end of each the study to further understand noticeable behaviors exhibited during the sessions. The average duration of the video tutorial used was 13 minutes 18 seconds (7:08 min, 14:48 max). The average duration of the sessions was 32 minutes 38 seconds (19:48 min, 40:32 max). Each participant was rewarded with a 15 USD giftcard.

We follow the recommendations for thematic analysis [11], and iteratively analyzed the interview data and the conversation logs three times in total with an interval of at least one day between sessions to enhance validity. Authors on our research team watched and open coded all screen recordings and think-aloud sessions. Then, the identified codes were reconstructed to the codes most relevant to our research questions through discussions. The codes were constructed around two themes: challenges, and opportunities of voice user interface in navigating how-to videos (Table 4.5).

Voice based interactions between users and systems can be seen as a type of conversation. To understand user strategies from their command utterances, we analyzed dialogue data between the user and the wizard using the turn-taking framework in conversational analysis [91].

4.4.1 Findings

Challenge 1 - Characteristics of How-to Videos.

Because of the sequential nature of the video (there is the concept of an **unknown future**), users often make a guess to navigate forward in the video, or they have to watch less relevant or less interesting segments. One illustrative example was when P2 asked the wizard *"could we change the speed to like 1.25? I want to slow it back down when she actually starts"*. Also, in the interview, P1 noted *"If I don't know what's coming up, I'm very uncomfortable skipping. If there's an outline, I would, but otherwise I don't know how much to skip or how much to speed it up by."* and P4 commented *"If I knew where I was going, I feel like I would progress better"* . From this we can conclude that it was difficult for users to anticipate what is coming up, and dealing with this uncertainty is an important design issue.

Challenge 2 - Voice Inherent Problems

When participants used a specific time interval for jumps, it often required multiple adjustments to navigate to the target even when the participant had a good sense of where the target was. In this case, **command parsing delays** become an important user interface limitation. P4 explained "saying go back by how much creates a delay between the point where I started saying the command (the point where I started saying the command) and when I finish the sentence and for you (wizard) to understand it. So I would have to say, for example, go back 30 seconds, and then go back 5 more."

4.5 Design Recommendation

Based on our findings and understanding from the three studies, we propose the following recommendations for designing voice based navigation for how-to videos.

4.5.1 Support Conversational Strategies

Support sequence expansions and command queues as both are strategies users often use. For example, supporting users to perform a single command multiple times in a row by recognizing "again" following "go back 5 seconds", and supporting users to place multiple commands in one utterance like "go to 2 minutes and 4 second mark and pause" would be useful.

4.5.2 Support Iterative Refinements of Commands

Users often need multiple tries to find the intended navigation target. It is because a) what users remember can be different from the part they are looking for or vice versa, b) sometimes users don't remember, and c) sometimes users remember but don't know the exact vocabulary like the names of knitting techniques and tools. Good examples are support for descriptive commands and keyword search in transcripts.

4.5.3 Support Interactions with User Context

Designing voice commands for how-to videos is not about supporting a single command, but understanding the higher level user intent behind the utterance is crucial. We identified all seven interaction intents (pace control pause, content alignment pause, video control pause, reference jump, replay jump, skip jump, and peek jump) that can be supported. One possible solution in distinguishing them is to set up the command vocabulary such that each intent has its unique keyword. For each of the intents, specific design recommendations are as follows:

Pace Control Pause & Content Alignment Pause

This is the pause for users to gain more time to finish the step. Keep a record of the point of pause for future references. Allow the user to easily compare the progress or the state of the user and those of the video by supporting various examination features like zoom or taking screenshots.

Video Control Pause

This is the pause where the user has an intention to navigate to other places in the video. Keep a pointer to the origin and provide "comeback" to this point, as it will often happen after jumps.

Reference Jump

Provide "memory". Augment users' memory to enable more accurate references by using features like markers and object annotations. Also, as reference jumps often happen in multiples, make the subsequent search processes easier, by suggesting updates or narrowing down of the interval of jumps.

Replay Jump

Support replay by allowing users to set a loop interval and the number of iterations.

Skip jump

Provide a visual summary of the remaining sections of the video for users to skip around. Approaches using instruction milestones, key frames, or frames containing user-specified keywords are all suitable.

Peek Jump

Provide a "comeback" feature to the origin position of the jump.

4.6 Discussion

Our study and interview results highlight the challenges, opportunities, and user expectations of using voice interfaces for video tutorials. We first discuss the user challenges of adapting to a VUI from a GUI when learning physical tasks with video tutorials. We then discuss how our research methodology of designing a series of experiments in progression can be extended to designing VUI for other applications and domains.

4.6.1 Transitioning from GUI to VUI

Mouse vs Voice

We found voice interfaces require an initial pause while issuing subsequent commands. For example, when using voice input in Study 2, users issued a pause command before every rewind command. In contrast, when using the traditional mouse interface, users never paused the video before skipping to another point. We think this is due to the time it takes for the user to speak the voice command and for the system to process it. Also, the target is directly specified with mouse (or touch) but with voice the target is often specified relative to the current position of the video. For example, if the user does not pause the video before jumping, the original reference keeps moving, and the interval they had thought of will not get them to the point they intended. As a result, the larger consequence is that voice-based interactions require more steps to achieve the same objective (i.e., pause + jump) than mouse-based interactions do (i.e., click).

Uncertainty from Unseen Content

When trying to navigate a video tutorial using voice, users make more concrete references to the past, whereas users have challenges describing later part of the video. For traditional video interfaces, scrubbing and clicking around are often used a solution to quickly peeking into the future. However, for voice interfaces, such a solution does not exist yet. Handling this uncertainty is an important design issue which would improve the usability of voice interactions for videos.

Recognition of Speech Input and Command Learnability

While the concept of using voice to navigate how-to videos is generally welcomed, participants also reported well-known problems of voice user interfaces. Speech recognition does not always work as expected, especially if users have accents or are in a noisy environment. In Study 2, nine participants also reported difficulty in figuring out the available commands. All participants showed frustration when the system did not respond to their command. Usability of VUI suffers due to relatively poor recognition, poor learnability and discoverability of available commands, and lack of feedback.

4.6.2 User Expectations and Opportunities

Multimodal Reference Strategies

Users often wanted to make references to the objects and speaker in the video. In Study 3, when users were making multiple corrections to navigate to a specific target point in the video, users have the advantage of utilizing the paused video frame as additional references, often employing **object references**. P1 explained "I look at the frame and the state of the objects that appear to see if it's before or after (the point I want to jump to)". Also, users often made **transcript references**, referring to things that the tutor has said. For example, P3 commanded the system "can you repeat that again? How she did multiples of four, the part where she said multiples of four". We believe voice assistants with a visual display could utilize this finding, as the referent needs to be visual or textual.

Conversational Strategies

Users often employ conversational strategies such as latent conversational intents and sequence expansions. Participants employed a lot of latent **conversational intents** frequently used in human-human conversations [91]. For example, participants said "Can I see it again, 10 seconds before?", "Can I see the last three knit?", and "Can you move it back to when she shows how it looks like from the back?". While a semantic answer to all of those questions would be a yes or a no, we contextually understand that these are requests, not participants asking for permission. Also, "I want to go back to the first time she does this with the second needle" by P6 is not a remark, but a command.

Participants often used **sequence expansion**, also heavily used in human-to-human conversations. For example, P4 said ("rewind 30 seconds until 3 minutes", "again") and ("slow it down to .5 and play from 4 minutes", "okay, from 3:55"). Users expected the wizard to have a memory of previous commands, and believed the wizard has the shared context.

Another strategy participants often used was including **command queues** in a single utterance. For example, P2 said "could we change the speed to like 1.25? I want to slow it back down when she actually starts the tutorial"

in the beginning of the video in the introductions segment. This requires multiple levels of understanding. The system would need to understand the first command to change the playback speed, detect when the tutorial starts, and remember to change the playback speed to normal. This is a powerful strategy that gives users more room to concentrate on the tasks by queuing multiple commands. P3 explicitly mentioned in the interview that "I want to sequence commands, set rules like if there is a second needle, slow it down." These techniques are applicable to generic voice interaction design, and existing services such as Siri and Google Assistant already support parsing "can I" questions as commands. However, all other conversational strategies described above is not supported.

Wizard Strategies

Users want "smarter" voice interactions that resemble a conversation with another human; the conversational agent that has complete knowledge of the viewing experience and can track progress. In Study 3, there are strategies participants used by relying on the "wizard" being another human. P6 requested "scrub the entire video" during the experiment and P4 noted in the interview "recognizing the repetition of commands like how you (wizard) did would be useful. If the system learned what I mean when I just say go back, and not having the description afterwards would be best".

4.6.3 Progression of Experiment Designs

In order to understand a user-centric design of voice interfaces for video tutorials, we carefully designed the three studies posing users in three scenarios in progression. Starting from how users use the current interface without voice interaction, to a basic adoption of voice interaction, to a Wizard-of-Oz interface with "ideal" voice interactions. We were able to create a taxonomy of current interactions, classify user intents in video navigation, and understand user challenges and opportunities for eliciting design recommendations.

We believe this progression of experiment design is generalizable to understanding how to design voice interactions for new applications or other domains like driving and exercising. For example, when understanding how to design voice interactions while driving, the same progression of studies could be just as effective. Understanding the current practices and needs of voice interactions while driving, and then using a design probe using a voice interface probe to understand opportunities and challenges, and then carrying out a Wizard-of-Oz study to elicit ideal interaction scenarios.

4.7 Limitations and Conclusion

One limitation of this work is that our design implications are observational with respect to 41 participants across three tasks. It is possible that other behaviors will emerge in tasks that have substantially different characteristics. Future work that analyzes voice commands at scale might be able to detect additional patterns.

Similar to how large-scale clickstream analysis can aid instructors to better understand learners' hurdles in watching online educational videos and reason about unsatisfactory learning outcomes [99], and to improve our understanding of interaction peaks and dropouts [54], we believe an at-scale analysis of voice interaction traces has potential to further our understanding on how to design better voice user interface. An an initial step, a live deployment of a voice-based interface for navigating how-to videos would be a worthwhile effort.

We also acknowledge there are other possible perspectives that we did not touch upon in the analysis. For

example, how navigation behavior and intents differ for novices and experts, and for first time videos and revisiting videos.

There are also practical issues related to implementing voice user interfaces that we do not address in this work. While speech recognition is rapidly improving, it is still far from perfect, and as observed in our experiments, speech recognition failures and delays cause user frustrations.

An additional technical challenge is related to audio source separation. In practical settings, audio coming from the video and possibly from the task itself may interfere with the user's voice commands, which would result in even poorer command recognition. While wireless headphones and earbuds are becoming more popular, there may be some situations where the user cannot use a dedicated headset.

Additionally, many ambiguities in users' voice command utterances that we discovered can be resolved by designing a system that understands and adapts to the *intent* of user and the *content* of the video. We believe determining these two variables in the wild is an interesting research challenge.

In conclusion, we present the first set of experiments that explicitly target voice based user interactions for navigating how-to videos of physical tasks. We examined how different user navigation objectives and intentions affect their word choices in voice command utterances, and reported a lexicon of types of interactions and the motivating factors behind these commands. Despite the limitations listed above, we believe that our experiments will be informative for researchers and practitioners who design voice-based video navigation systems, which have the potential to play a large role in how learning systems of the future operate.

Chapter 5. RubySlippers

5.1 Introduction

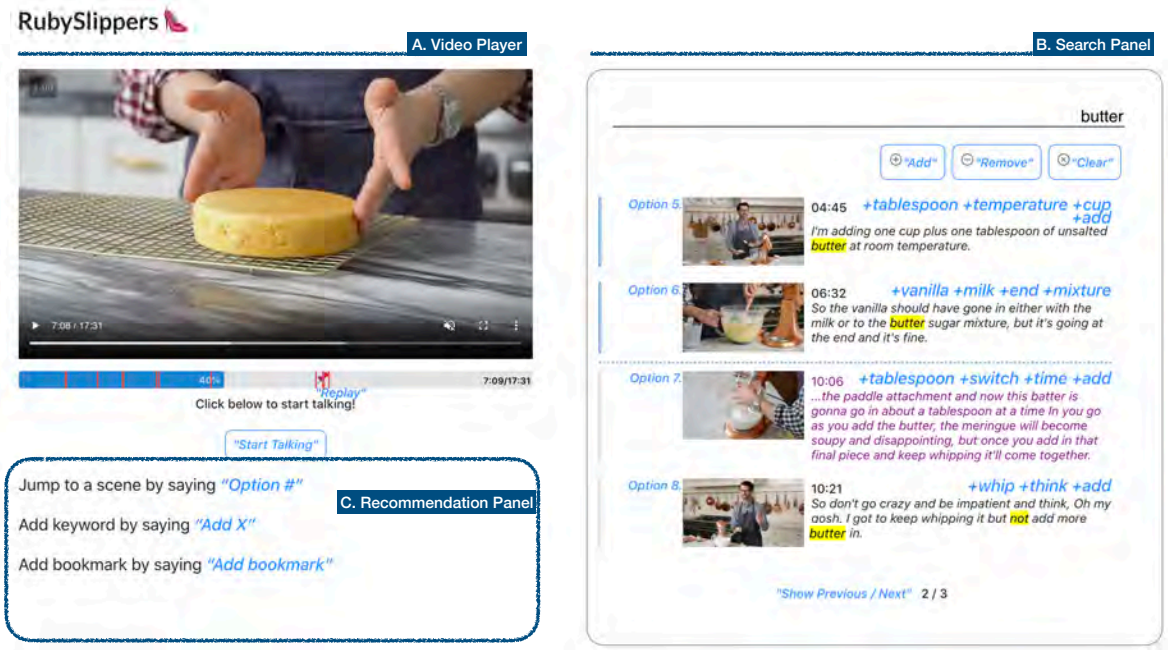


Figure 5.1: RubySlippers is a multi-modal interface supporting voice navigation with three main components: (A) Video player and timeline. (B) Search panel where keywords-based search results are shown. (C) Recommendation panel provides suggestions of search keywords and available navigation commands at each interaction interval.

Learning from how-to videos involves complex navigational scenarios [18]. For example, people often revisit previously watched segments to clarify misunderstandings, skip seemingly familiar contents, or jump to the later parts to see the result and prepare for future steps. A common strategy people use in traditional video interfaces with mouse and keyboard is "content-based referencing"—for example, peeking the video's content either by scrubbing or hovering on the timeline for thumbnails or performing sequential jumps to move the playback positions.

However, how-to videos for popular tasks, such as cooking, makeup, and home-improvements, require manipulations with physical objects and involve physical activities. As a result, viewers need to use both hands to control the video and carry out the task at hand. This incurs costly context switches and heavy cognitive load while tracking the progress of both the video and the task. Ideally, voice user interfaces like Alexa or Google Assistant can provide an opportunity to separate the two activities - controlling of video using voice and applying the instruction with two hands.

The most straightforward and standard method of supporting voice interaction for video interfaces is to directly translate the timeline manipulation into voice commands, such as navigating the video with commands

like “skip 20 seconds” or “go to three minutes and 15 seconds”. However, this “temporal referencing” strategy requires a different mental model for navigation than directly manipulating the timeline because it limits the users’ ability to peek into the content.

With a formative study, we have identified four challenges for supporting “content-based referencing” in voice user interfaces for how-to videos. First, users want to use succinct keyword-based queries instead of conversational commands to alleviate the burden of constructing sentences. Second, users cannot recall the exact vocabulary used in the video. Third, users have difficulties with remembering the available commands. Finally, unlike timeline interactions, voice inputs suffer from speech recognition errors and parsing delays.

To overcome these challenges, we present RubySlippers, a prototype system that supports voice-controlled temporal referencing and content-based referencing through keyword-based queries. Our computational pipeline automatically detects referenceable elements in the video and finds the video segmentation that minimizes the number of needed navigational commands. RubySlippers also suggests commands and keywords contextually to inform the user about both available commands and potential candidate target scenes.

In a within-subjects study with 12 participants, we asked participants to carry out a series of representative navigational tasks, focusing on evaluating the effectiveness and benefits of content-based referencing strategies. Participants found the keyword-based queries—our main feature for supporting content-based referencing—in RubySlippers useful and convenient for navigation. They were also able to effectively mix content-based referencing and temporal referencing using RubySlippers to fit the needs of their navigational task.

This paper makes the following main contributions:

- Results of analysis comparing temporal based referencing and content-based referencing techniques in voice video navigation. Specifically, challenges with content-based referencing in voice user interfaces.
- RubySlippers, a prototype video interface which supports both temporal referencing and content-based referencing with voice
- The computational pipeline that segments a how-to video into units that effectively support keyword based interaction techniques for navigation
- Findings from user studies, what findings

5.2 Formative Study

In this research, we characterize two navigation strategies, temporal referencing and content-based referencing. Temporal referencing is when the anchor of navigation in the user mental model is the time. For example, in a typical GUI-based video player, viewer uses temporal referencing by clicking on the timeline when the viewer knows that’s exactly the timestamp of the targeted scene. For voice user interfaces, the voice commands like “skip 20 seconds” support temporal referencing.

For content-based referencing, the anchor for navigation is the content. For example, viewers use content-based referencing when they examine the thumbnail or moves around playback position of the video to navigate to the target scene. Translating this to voice user interfaces, users must be able to issue voice commands that describe the content of scenes like “go to the part where the chef dices tomatoes”. Although for the latter, we are yet to see a voice-driven system design that supports this effectively.

To understand the advantages and disadvantages of the two referencing strategies in voice user interfaces, we conducted a formative study with 12 participants.

5.2.1 Research Probe

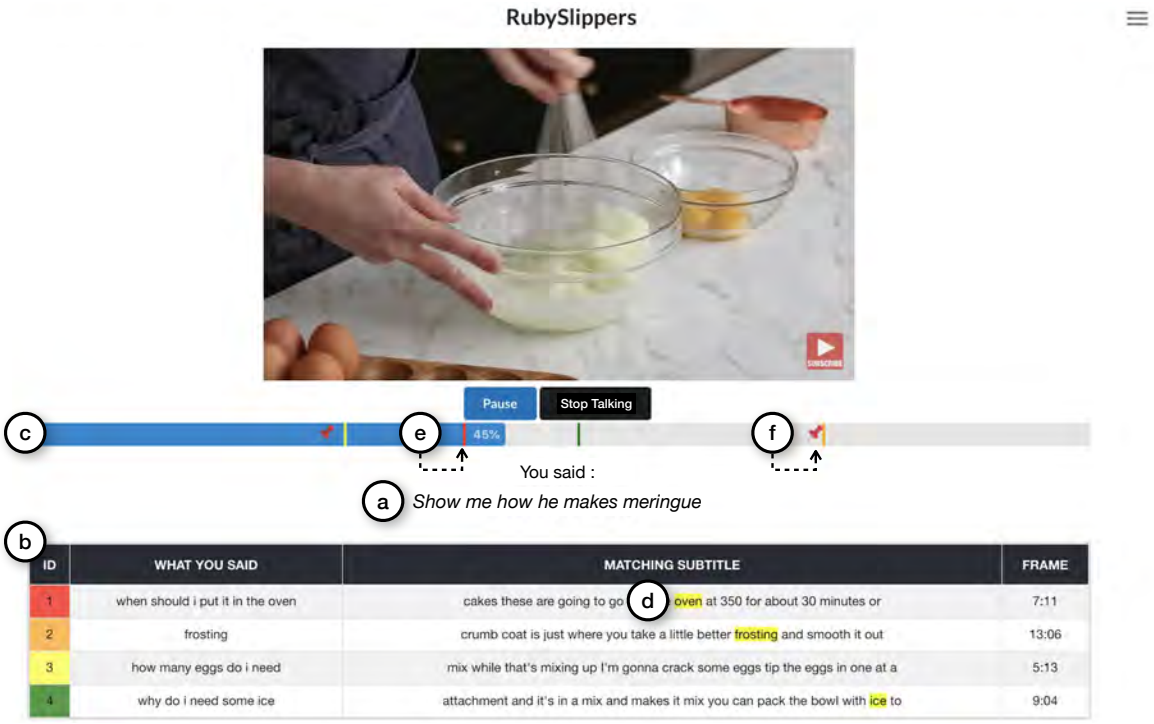


Figure 5.2: Our research probe supports temporal referencing and content-based referencing through basic speech recognition. Main features of research probe: (a) Real-time transcript of the user is shown. (b) The history table stores all previous queries, matching subtitle and the timestamps and highlights the word with the most influence to the resulting target scene. (c) On the progress bar, the scenes resulting from previous navigations are marked (e) which users can easily reissue with shortcut. (f) Users can also manually add bookmarks.

As the apparatus of the formative study, we built a voice-enabled video player that supports both temporal referencing and content-based referencing (Figure 5.2). Using the player, users can play, pause, fast-forward, and rewind by specifying the location or the interval for temporal referencing. For content-based referencing, users can describe the target scene they are looking for in a conversational manner. The system parses the users' voice query and calculate the similarity against each sentence in the transcript for the best match. We used word-mover distance [**?**] based algorithm for matching. While the method used not being the state of the art technology, we judged the accuracy this method provided was enough to understand the advantages and challenges of content-based navigation at this stage. Considering the frustration from the speech recognition errors, which was well reported in [76], we asked our participants to look beyond the speech recognition errors. The research probe also stores all previous queries and the matched results in a table (Figure 5.2.(b)) as a bookmark, which users can simply refer to by their ID to reissue the same navigation query, like "go to 2" for second item in the bookmark.

	Mental	Physical	Temporal	Performance	Effort	Frustration	$\overline{X}$
Temporal Referencing	5.67	4.67	6.25	5.58	6.83	6.67	5.56
Content-based Referencing	6.07	4.21	5.86	5.29	6.36	5.86	5.61

Table 5.1: Cognitive load measured with NASA-TLX for 12 participants. There aren't any significant differences in cognitive load between the two referencing strategies.

5.2.2 Study Procedure

We recruited participants with an online community advertisement. The criteria for invitation was the prior exposure to video tutorials and basic English proficiency.

We conducted a counter-balanced within-subjects study, where participants could only use one of the two strategies to perform five common navigating tasks in how-to videos: navigation to a scene with specific object usage, navigation to all scenes where a specific object appears, navigation to a scene with factual information, and multiple video comparison. To evaluate the efficiency and efficacy of each strategy, we measured both task completion time and cognitive load with 10-point scale NASA-TLX [**?**].

At the end of the study, the participants were given the freedom to explore mixing the two strategies. We also conducted semi-structured interviews to gather qualitative feedback for a deeper understanding.

To give a preview of the system, we gave a brief tutorial session on how to use each feature. To familiarze and build trust in the system's ability, we also provided some example "working cases", like "play", "paust", "stop" for temportal referencing, and "show me where she bought potatoes", "How many liters of water does she use for the plant" for content-based referencing. There were 3 sessions altogether and 3 domains of how-to videos were chosen: baking, packing and planting. In the first two sessions, participants were restricted to use only one navigation scheme where in the last session they were allowed to used both. In each session, 5 questions from 3 different types were asked to the participants. The first type was visual search, which meant that they were asked to navigate to the frame. Both single target and multi target questions were asked. The second type was video question and answering. Here we asked both short answers and long answer questions. In the third type, multi-video search, participants were asked to watch 5 different videos of the same domain and answer the questions which asked about the trend of watched videos. To complement the performance analysis with a qualitative understanding of participants' experience, we included semi-structured interviews. Each interview session took about 15 minutes asking about their how they felt about using the system.

5.2.3 Results

The difference in cognitive load between the two strategies is not significant. People reported a high task load in both the temporal and content-based referencing Table 5.1. However, the finding we learned is that the sources of task load for each strategies are different.

For temporal referencing, participants said it is tedious and laborious to jump around because in this condition they had to specify the exact temporal location of the scene that they want to watch, and remembering exactly when a certain event happened or having to make multiple corrections to reach the moment in the video is very tiring. They also felt pressured.

For content-based referencing, participants' stress mostly came from system failures in understanding their utterances. Participants were thinking too much about which words they should pick when formulating the query

sentences, because either they could not remember the vocabulary or because they wanted to be efficient and find one magic word to include that makes the hit. From the interviews, participants reported issuing a voice command in "conversational" form is burdensome, and would rather use a combination of discrete keywords. For example when P5 tried "From which shop did she buy the bags?", the system did not populate the scene P5 wanted. In the interview, P5 said "I had to try hard not to include words that are less necessary, but buying and bags ARE necessary. It's so stressful to come up with a *correct* sentence, and repeat long sentences over and over." Participants felt like there is one correct sentence that will take them to the scene they want, and it suddenly became a guessing game for them that they did not want to play.

From the interview feedback, we deducted the problem of content-based voice navigation into the problem of how to help users find the minimum set of keywords that describe the scene they are looking for. Combining both study results and the interview findings, we have identified the following user challenges in efficiently navigating how-to videos difficult:

C1. Difficulty in referring to objects and actions that appear multiple times across the video

C2. Difficulty in precisely recalling the exact vocabulary due to divided attention

C3. Difficulty in remembering what the available commands are and how to execute them

C4. Inconvenience caused by the time delay from parsing and speech processing

The first challenge is that the same objects and actions appear multiple times throughout the video, and the more important they are, the more frequently they appear. This directly conflicted with what participants wanted to do. Participants wanted use as fewest words as possible when referencing. We observed that especially to minimize parsing errors, participants tried to use shorter and shorter sentences when they were experiencing system failures. However, because the objects and actions appear in multiple places across the video, participants needed to construct longer sentences in order to narrow down, which caused more parsing and recognition errors.

The second challenge is that participants have difficulty with recalling the exact words used in the video because the attention of the user is divided into performing the task and formulating the query. While participants noted recall of the words as easier than recall of the timestamps, it is still challenging especially when equipped with little background knowledge. For example, when participants were presented with an image of "a carry-on"—the precise term used in the video— and were asked to "find at which shop the person in the video bought this?", they tried different words like bags, baggage, luggage and suitcase in their query sentences. The system could not find the correct scene. Also, P2 first searched for the word "sugar" to find out how much was needed. When 17 results showed up, P2 tried with the query "spoon of sugar" but got 0 matching result. Then, P2 tried with the query "cup of sugar" and got 10 results. P2 failed in narrowing down the search, and had to examine all the options to find the answer.

The third challenge is that users do not know what commands are available nor how to execute them. Users are frustrated when they forgot how to initiate commands or update them when the initial command failed. Participants repeatedly asked how to talk to the system, and whether they can see the list of commands next to them all the time.

The fourth challenge is that voice interactions take more time, because they have parsing delays whereas direct manipulation of the timeline, which most users are already accustomed to, does not. The first two challenges

are cause by the characteristics of a video tutorial and the latter two are commonly reported challenges in voice interfaces.

The first two challenges were uniquely identified through our study, where the latter two are well-reported in previous research in voice interaction usability.

5.2.4 Design Goals

Based on the analysis of the interview and suggestions from the participants, we identified three design goals for tools to support content-based voice navigation for how-to videos. The design goals individually address three key user tasks in voice based video navigation, which are initiating command (D3), referencing (D1), and revising the command (D2).

D1. Provide support for efficient content-based referencing using keywords rather than full sentences.

D2. Provide support for effective query updates.

D3. Provide support for informing users about executable commands and potential navigation.

5.3 RubySlippers

With the three design goals in mind, we present RubySlippers (Figure 5.1), a voice enabled video interface that allows users to use both temporal referencing and content-based referencing. Below, we walk through two scenarios illustrating some of the advantages of using RubySlippers when navigating how-to videos, and subsequently describe the features that enable content-based voice navigation. We then also describe the computational pipeline that powers RubySlippers.

5.3.1 Scenario 1

Dorothy loves to cook at home, but is a novice at baking. She wants to make a birthday cake for a friend with the help from a video tutorial online. She decided to use RubySlippers to avoid touching the computer with hands covered in flour. For the first couple of minutes, she easily followed the instructions using pauses and by changing playback speeds with voice. However, when the chef in the video put the vinegar into the mixture, she couldn't remember how much vinegar was needed. As preparation of the ingredient was in the earlier part of the video, she talked to the system "Vinegar" and could easily find the scene where the Vinegar is being added on the search panel. Dorothy had to just say "option one" to navigate to the part.

While Dorothy was busy whipping the cream, the video kept playing and moved on to a few minutes later. After a couple of failed attempts to guess the original location with the command "Go back 30 seconds", she talked to the system "Cream". However, RubySlippers displayed more than ten scenes where the word "cream" was mentioned. Instead of peeking into all the options, she simply added "whip" by saying "add whip" and came back to the original point.

5.3.2 Scenario 2

Glinda, a friend of Dorothy, is throwing a birthday party tonight. She is preparing for a party makeup and selects a how-to video of her style. While it is her first time using RubySlippers. After the lip makeup, she wanted

to skip the step of blushing cheek and watch how to do contouring. When she said “Contour” to the system, it responded with a list of synonyms appearing in the video which were “Bronzer, Outline, Brown, Shadow, Darken". So she replaced her query with “Bronzer” and could quickly reach the target scene.

Glinda was following the step of applying the glitter on her eyes. While she was applying it to her right eyelid, the video—edited to avoid redundancy—fast-forwarded the same process with the left eye and moved on to the next step. After she re-visited the same scene multiple times to finish the left eye, RubySlippers automatically added a “Replay" mark, reducing the burden of Glinda to repeat the query. When she did a couple of more jump-back by saying “Replay", a loop was created which repeated the same step with no input until she escaped.

5.3.3 Keyword-based Querying

To address D1, RubySlippers supports keyword-based queries for users to describe parts of the video they would like to navigate to. These keywords are pre-populated using an NLP pipeline which we later describe in the 5.3.7. RubySlippers returns the list of scenes resulting from the keyword-based search below the search bar Figure 5.3.(b). The corresponding locations on the timeline are marked with vertical orange lines (Figure 5.3.(b)). The search keyword is highlighted in the transcript corresponding to the scene.

Figure 5.3: RubySlippers Search Panel: In the second component, users can search and choose among the option scenes. (a) Users can update the current query by adding, replacing or removing keywords. (b) The search result is shown in chronological order. Each item has a visual thumbnail, timestamp, transcript and keyword suggestions for further query specification. (c) Users can also browse search result pages with voice commands. (d) Keywords that help users narrow down the search result are shown.

5.3.4 Updating Queries with Keyword Composition

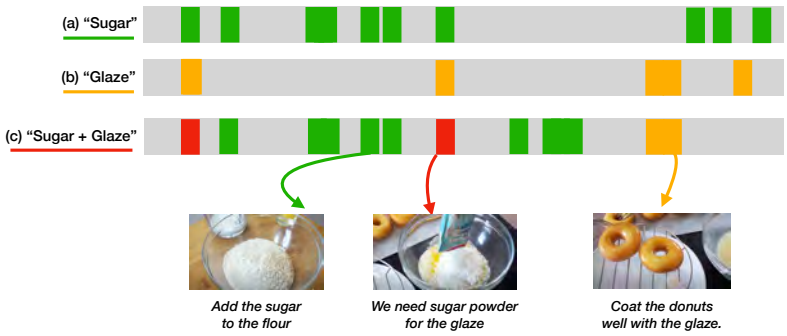


Figure 5.4: An illustration of how query updating with keyword composition works. Parts containing the keyword “Sugar” are marked in green, parts containing the keyword “Glaze” are marked in orange. The part that contains the composition to two keywords “Sugar” and “Glaze” are marked in red.

To address D2, users can update their previously issued query by adding, replacing or removing keywords (D2). (Figure 5.3.(a)) RubySlippers assists this query update by informing the users which keywords can narrow down their search result when added to the current query (Figure 5.3.(d)). For each search item in the list, keywords that are likely to uniquely describe the item while also reducing the number of search results when added are shown. A visual illustration of how query updating works in shown in Figure 5.4. Also, there is a vertical bar in the search results which indicates where the current video playback is. The visited scenes are visually distinguished with different color so that the user can quickly understand which options are unseen.

5.3.5 Command and Keyword Suggestion

To address D3, RubySlippers displays available commands or example keywords for initiating the navigation (Figure 5.5). RubySlippers takes both the current user state and the previous interaction into account to make recommendations. For example, the system shows a word cloud [**?**] for initial seed (Figure 5.5.(a))and displays available commands like “undo" to recover after a navigation has been made. It also suggests semantically similar words if the input keyword from the user does not appear in the video (Figure 5.5.(e)) and helps users to make quick search by informing how to narrow down the number of options when there are too many (Figure 5.5.(c)).

5.3.6 Automated Bookmarks

RubySlippers creates an automated bookmark for frequently visited scenes. Users involved in physical tasks frequently pause to control the pace of the task and to make sure the task progress is aligned with the video’s progress [18]. So users often need to visit the same spot in the video multiple times. With the automated bookmarks, users can revisit a previously visited scene without issuing the same command repeatedly. People can avoid recalling their previous queries, which is cumbersome and difficult with voice, but rely on recognition [**?**].

When a same referenceable unit is visited more than two times, RubySlippers automatically added a “Replay" mark on the timeline (Figure 5.6.(c)), reducing the burden of users to repeat the same query. In Figure 5.5.(d),

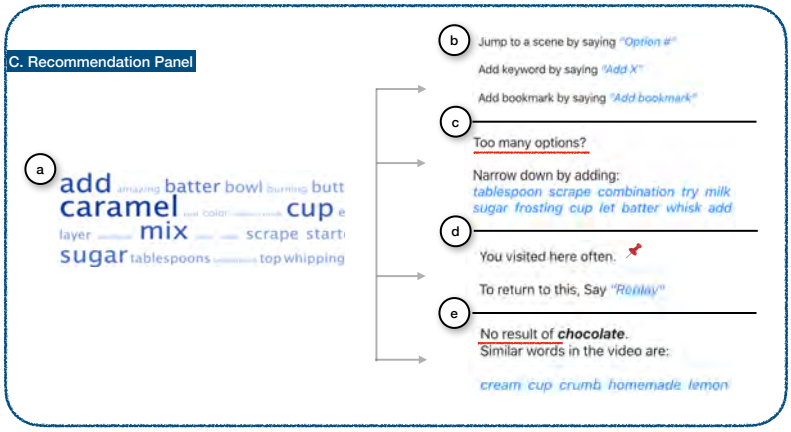


Figure 5.5: RubySlippers Recommendation: The third component provides recommendations which adaptively change at each interaction interval. (a) The system displays a word cloud for initial seed. It serves as a keyword summary that can help users recognize and remember the main events happened in each video. (b, d) When the user starts the navigation, it shows available commands so that users can smoothly connect to the next interaction. (c, e) Keywords that can be added to the current query are suggested in support of users narrowing down or fix the search.

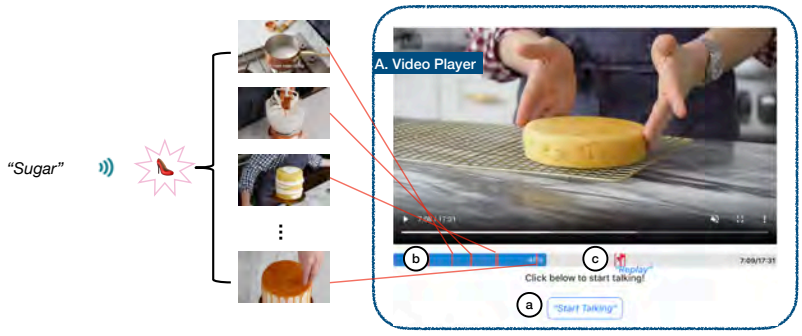


Figure 5.6: RubySlippers Video Player: The first component consists of the how-to video and the timeline bar showing the progress of the video, and the speech recognition status is shown below. (a) To start speaking to RubySlippers, users first must turn on the speech recognition by clicking the "Start Talking" button. After the recognition is on, real-time transcript of the user is shown. (b) On the timeline, the timestamps of the search result items are marked with vertical red lines. (c) Bookmarks for frequently visited scenes are automatically created and marked with “replay”.

RubySlippers inform users of the creation of bookmark and how to use it. When there are more than one bookmarks, each is specified with the bookmark number to which users can refer to distinguish one from another.

5.3.7 Computational Pipeline

The computational pipeline that powers RubySlippers segments the transcript into units of that contain referenceable keywords to support keyword based querying and query updating. We highlight its three components: 1) video segmentation, 2) option population, and 3) keyword suggestion. First, the pipeline pre-processes the

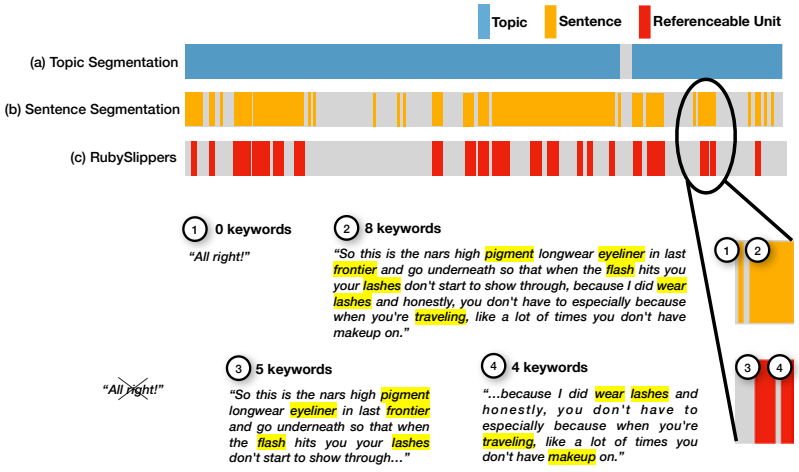


Figure 5.7: Our pipeline segments the transcript into units where the number of keywords are balanced. This pipeline is designed to make keyword-based queries efficient, in that more fine-grained searches are possible, and narrowing down the search with adding keywords is faster.

transcript of the video and runs a part-of-speech tagger to pre-populate proper nouns, nouns, and verbs which correspond to objects and actions. The intuition is that they likely correspond to objects and actions, which the users can reference in navigation. Both the observations from our formative study and the prior work in cognitive psychology [**?**] show that people organize knowledge structures about an event around object and action as units.

We then split the video transcript into units which users can refer to with keywords and are in lengths containing no more than five keywords, making it easier to be understood at a glance. First, we use sentence-level segmentation using punctuation marks.

To recover from punctuation errors in the transcript, we run the BiRNN punctuator [**?**] over transcripts and further fix the punctuation marks. Two of the authors examined the resulting transcripts for corrected punctuation to further enhance the validity. Since the text in the transcripts in these how-to videos are informal and colloquial, the sentences are often incomplete sentences and grammatically incorrect. Therefore, off-the-shelf sentence segmentation techniques yield segments that are longer than typical sentences we expect. Thus, we use dependency relations to further split those that have more than 50 tokens. We run a dependency parser to find "conjunction words", and split long segments by using these conjunction words as delimiters.

Then we take each of the "clauses", count the number of "keywords" in it and split this long "clauses" into "referenceable scenes" where each scene contains at least two "keywords" but no more than five keywords.

To summarize, our resulting "referenceable scenes" may or may not by full sentences, and the segments might even be in the middle of a "sentence". The output of our method is illustrated in Figure 5.7.

For populating the search result, we use exact keyword matching. This means only the ones that containing maximum number of keywords in the query are populated and returned as search results. For populating the additional keyword suggestions in each scene for assisting the query update, we select and show the ones that significantly decrease the number of option scenes. If the query results have more than 12 scenes, adding one of these keyword suggestions will drop the number of scenes below 12. If the query results contain less than 12 scenes, then adding one of these keyword suggestions will drop the number of scenes below 4.

Participant Number	Age	Gender	Prior Experience Watching How-to Videos		
			Makeup	Baking	Others
P1	19	F	YES	YES	Home Workout
P2	20	M	NO	NO	Arts and Crafts
P3	23	M	NO	NO	Programming
P4	25	M	NO	NO	X
P5	24	M	NO	YES	Arduino
P6	24	M	NO	YES	Piano
P7	27	M	NO	YES	PC Assembly
P8	23	F	YES	YES	Dance
P9	20	M	NO	NO	Guitar
P10	24	F	YES	YES	X
P11	24	M	NO	NO	Juggling
P12	20	M	NO	NO	Piano, Programming

Table 5.2: Background Information of Study Participants

5.4 Evaluation and Results

We evaluated the effectiveness and task load of RubySlippers through a lab study. Speficically, goals of our evaluations were (1) to assss the feasbility of keyword-based querying and updating as means of supporting "content-based navigation", (2) to assess the effect in task load the addition of "content-based referencing" to "temporal referencing", and (3) to gain feedback on the effectiveness of RubySlippers in the following three representative navigational tasks.

1. single target navigation for information seeking
2. multi target navigation for information seeking
3. following along the video while applying the instructions to the task at hand

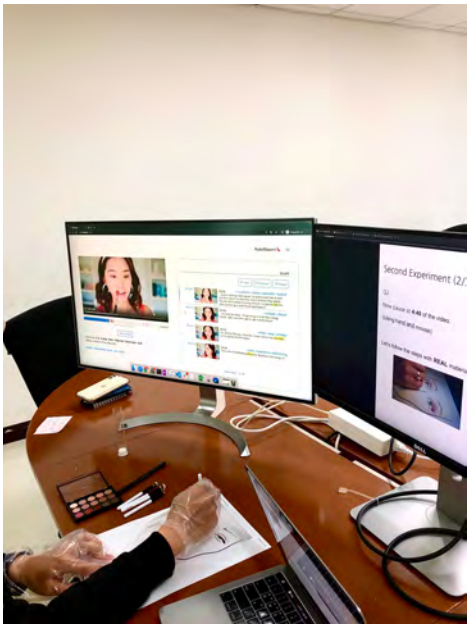
We conducted a counterbalanced within-subjects study between, navigational support(temporal vs temporal + content-based), video domain type, and the order.

5.4.1 Participants

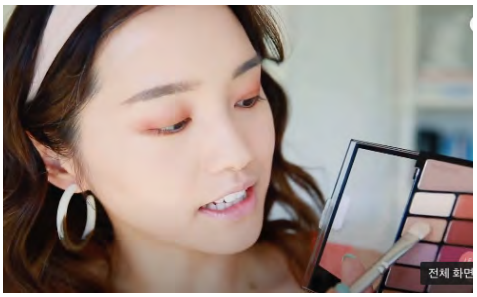
We recruited 12 participants (9 male, 3 female, mean age 22.75, stdev=2.45, max=27, min=19) through an online community posting. People who participated in the formative study were excluded from this recruitment. Each study session was 80 minutes long, and the participants were paid 20,000 KRW (~USD 17).

5.4.2 Study Procedure

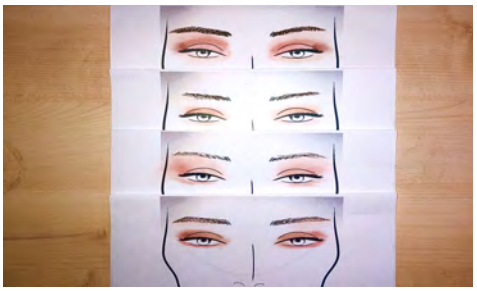
We followed the safety guideline [**?**] to ensure safety during COVID-19. When participants arrived at the experiment location, they were asked to measure their body temperature and wear a mask. They were asked to sanitize their hands and wear gloves throughout the whole study process. They were asked to bring their own headset. All participants' body temperatures were within the normal temperature range. During the experiment,



(a) Experiment Setup



(b) How-to Video



(c) Follow Along Physical Task

Figure 5.8: Participants were instructed to use only voice to control the video during tasks.

ID	Title (Duration)	Domain	Creator	URL
Video 1-1	Amazing Caramel Cake(1051s)	Baking	Preppy Kitchen	[?]
Video 1-2	Amazing Hot Cross Buns(1083s)	Baking	Preppy Kitchen	[?]
Video 2-1	Drugstore Makeup Tutorial(926s)	Makeup	Jenn Im	[?]
Video 2-2	Passport Photo Makeup(936s)	Makeup	Roxette Arisa	[?]

Table 5.3: Four how-to videos used in the evaluationi study.

all windows were open and air conditioner was turned on to keep the room ventilated. We sanitized the room after each session.

After the safety check, participants were given a tutorial of the RubySlippers interface, and they were asked to go through a practice session to familiarize with the interface. Then the participants were asked to complete navigation tasks in two sessions. In each session they are assigned different domain of how-to video - one of baking or makeup - and different navigation condition - either temporal reference only or both temporal and content-based approach. Participants were instructed to use only voice to control the video during tasks. During the experiment, two different monitors were shown to the user (5.8.(a)): one showing the system, RubySlippers and the other displaying the task at the moment. We ran semi-structured interviews after the session to gain a deeper understanding of the strengths and weaknesses of RubySlippers and the decisions about specific referencing strategies participants employed, or interesting usage patterns. We recorded participants' screens and audio during each session upon their consent.

We selected four single-person video tutorials on YouTube shown in Table 5.3. The criteria for selecting videos include duration of the video, amount of information given verbally, and the availability of manually added captions. There were two sessions per participant and two domains of how-to videos were chosen: baking and

Video Type	Task Type	Time to Complete (sec)	
		Temporal	Temporal + Content-based
Video 1-2	1-a. Multi-Target Search	230	310
	1-b. Frequently Appearing Object Search	240	185
	1-c. Follow Along Physical Task	545	525
Video 2-1	2-a. Frequently Appearing Object Search	270	295
	2-b. Follow Along Physical Task	365	400
Video 2-2	2-c. Frequently Appearing Object Search	335	230

Table 5.4: Median time-to-completion in seconds per video per navigation task

makeup. For each domain we selected two videos. One how-to-bake video (Video 1-1 of Table 5.3) was used only for the practice session. Video 1-2 was used when the condition of the session was baking and both Video 2-1 and 2-2 were used for makeup condition. We chose two makeup videos instead of one because while all four videos are in almost the same length, the amount of transcript of the instructor varied such that the amount of textual information in two makeup tutorials were almost equivalent to that in one baking tutorial.

In each session, three different tasks were given to the participants. The first task type was a multi-target navigation task, in which participants were required to navigate to multiple scenes to find all the answers. For example, participants were asked to find answers to the question "how many and what type of brushes are used in the video?". Second task type was a single-target navigation task, in which we asked information about a frequently appearing object in the video. For example, participants were asked to find and explain what the chef does to expand the dough, while "dough" appears 16 times in the video. In the third task type, physical task, participants were asked to follow along some part of the video (about 90 seconds) with real ingredient and tools prepared by the researchers. For example, they were asked to apply eye makeup on the printed face on a sheet of paper. We prepared ingredients and tools needed to follow the cooking videos, and also makeup supplies needed to follow the makeup video. Participants were wearing plastic gloves at all times.

5.4.3 Results and Findings

We summarize the quantitative results and present main findings with respect to the three design goals, usage patterns, and usability and usefulness of RubySlippers.

Quantitative Results

The results of the time taken for task completion and the number of interaction measurement in median are depicted in Table 5.4 and Table 5.5. A Wilcoxon Signed-Rank test was conducted to evaluate the effect of type of reference on the time taken and number of interaction made to complete the task. For all tests, an alpha level of 0.05 was used. There was no significant reference type effect.

We did not observe any significant median difference in time taken for task completion between the two conditions. The results are in Table 5.4

For the number of interactions, we counted the number of command invocations participants made. We counted repeated invocations of the same command due to recognition failure as one invocation. Even though

Video Type	Task Type	Number of Interactions	
		Temporal	Temporal + Content
Video 1-2	1-a. Multi-Target Search	12	6
	1-b. Frequently Appearing Object Search	5	7.5
	1-c. Follow Along Physical Task	18.5	16
Video 2-1	2-a. Frequently Appearing Object Search	15	11.5
	2-b. Follow Along Physical Task	14	14
Video 2-3	2-c. Frequently Appearing Object Search	11	7.5
Total	Temporal: $(\overline{X} = 13.33, max = 29, min = 4, stdev = 6.44)$		
	Temporal + Content: $(\overline{X} = 10.33, max = 22, min = 2, stdev = 4.89)$		

Table 5.5: Median number of command invocations per video per navigation task and the summary of each condition

there aren't any statistically meaningful differences between the conditions, there is a tendency that people issue less commands in the "temporal and content" condition as shown in Table 5.5

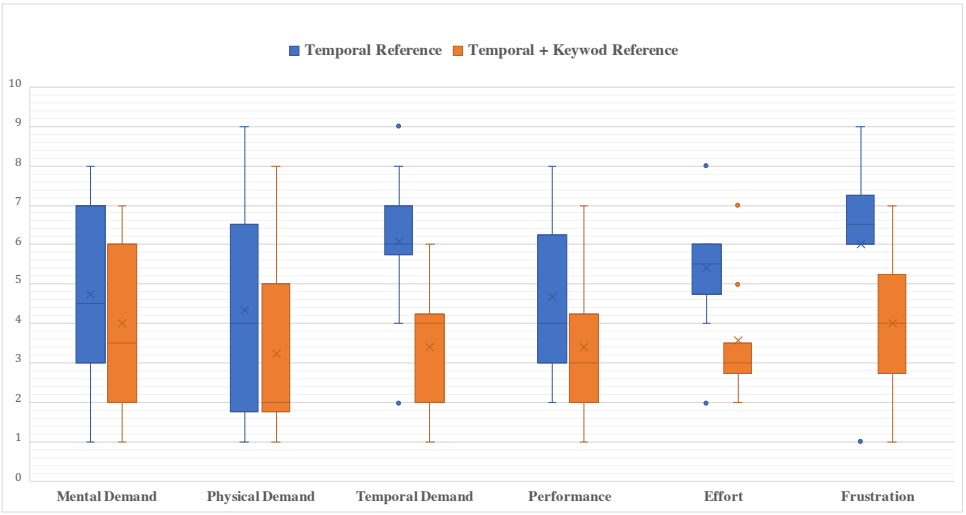


Figure 5.9: Boxplot of median NASA-TLX scores for the 2 reference conditions (0 = low, 10 = high)

Figure 5.9 gives the box plot of the NASA-TLX scores using a median. We saw a statistically significant drop in temporal demand, effort, and frustration in the "temporal and content"condition. While not as significant, effort has also decreased in the "temporal and content"condition.

Qualitative Feedback

The feedback from participants was overall positive, with all participants agreeing that RubySlippers is useful for navigation. After having experienced both conditions, participants appreciated the ability to concisely express their intended navigation target using keywords. P11 said *"I could definitely see how it would be helpful. Especially having experienced content-based referencing first and then when I had to use temporal referencing only, I kept feeling the urge to use content-based referencing."*

D1. support for efficient content-based referencing using keywords Participants found that content-based referencing is more robust in information seeking tasks. Participants in temporal referencing only condition had to rely mostly on visual cues, but missed many relevant timings and information while skipping around and fast-forwarding. Five participants had explicitly said that they're not sure if they had found all the information they were asked to find. On the other hand, participants in "temporal and content"condition were confident that found all the answers, and that they trust the system more. Specifically, P4 said, *"I can focus more on tasks not having to keep memory of timestamps nor the overall order of some independent events. I can use less commands to find what I want, and it leads to less delays! Apart from the (speech processing) delays and recognition errors I love the idea!"*

D2. support for effective query update Participants found the idea of using composition of keywords for updating queries convenient, and that it helped them understand the content better. P6 had noted, *"It's useful to be able to search the combination of multiple keywords - not just for narrowing down the search but I could see how to objects are used together."* For example, when butter and cream are used together salt is also being used. While it was enough to use butter and cream to find the relevant information, participants also learned salt is another important ingredient used in the step. Also, the participants found the search result items shown on the search panel on the right as structural dividers, and found it useful for navigation. P10 had said, *"It's just like video chapters (on description or command on YouTube) but in micro levels. I like it as it's simple and work as a shortcut."* The search results are the parts of the video that queried objects and actions appear. Participants found these to be meaningful markers that helped them understand whole task procedure of the entire video.

D3. support for informing users about executable commands and potential navigation Participants also found the ambient help of displaying the available commands and potential keyword suggestion useful. P8 said,*"I sometimes said keywords that doesn't exist in the video, but I was able to quickly recover by looking at the keyword suggestion. It was super useful."* Similarly, P5 noted,*"Even when I cannot recall or do not know the exact name of the object, I searched for an action related to that object or a co-occurring object. Looking at the keyword suggestions in the options list, I could figure out the name and go to that scene!"*

Application to other types of how-to videos When asked what other types of how-to videos they think RubySlippers would be useful for, participants showed excitement and provided many interesting ideas. Many participants said long videos that they do not have the patience to watch would benefit from RubySlippers even if it is not a tutorial video.

Two participants mentioned how RubySlippers would be useful for lecture videos. P4 noted, *"I want to use this system for watching lecture videos when my hands are tied from note-taking. When the formula taught in earlier part of the video is used later and I cannot remember the exact equation, I can easily navigate backward and return."*

Participants noted that videos that have less clear structure would also benefit from RubySlippers. P1 said, *"Unlike baking or makeup which have strict orders of steps, it's hard to guess the time of random-order videos (home workout). So in those, it will be more useful!"*

Usage Patterns Four participants used content-based referencing for navigating to a general area and temporal referencing for refinement when they were looking for an exact moment in the video. This is quite effective

because the user is narrowing down the search space using keywords, and then pin pointing the exact scene with specific timestamp.

Also, participants used groups of results that are closed together on the timeline as a unit of navigation. For example, in the make-up video, a number of brushes appear. When searching for a specific brush in the video, participants used one word query "brush", and hypothesized the resulting scenes that are closer together are likely using the same brush, so they would examine one scene from each group to find the brush they waned.

The most interesting and unexpected finding is that P12 found content-based referencing helpful for understanding and learning the tutorial content. P12 had said, *"It would also help me to memorize the content of the video - Keyword-based referencing helps me to remember keywords and key concepts of the video much more than temporal-referencing."* This is particularly interesting, and warrants further investigation and future work, because we might be seeing effects of "self-explanation" [**?**] as a byproduct of efficient interaction design for voice navigation.

5.5 Discussion, Limitation, and Future Work

We discuss findings, generalizability, and possible limitations of this work.

5.5.1 User Confidence and Trust in Content-based Referencing

Participants reported they felt more confident when navigating and trusted the query results more when using content-based referencing than using temporal referencing. We hypothesize this is a result of our pipeline using exact keyword matching: the results are binary, either the keyword is in the search result or it is not in the search result. Scenes shown in RubySlippers are guaranteed to contain the keyword, which helps participants build a clear mental model of what to expect from a query result.

5.5.2 Beyond text-based content

To see how our approach can generalize to other types of how-to videos, we performed a breadth-first random sampling of the popular how-to video domains in order to classify them according to a variety of objects and actions appearing in the video.

We explored top three hundred video tutorials viewed the most when searched with the query "how to step by step" on YouTube and distilled 17 different domains of how-to video involving physical activity. For each domain of how-to videos, we examined three videos with similar length (around 10 minutes) and averaged the number of objects and actions appearing in these videos while regarding body parts as objects.

In the proposed classification Figure 5.10, the styles of how-to videos are shown with respect to the two dimensions: object and action. The classification is done qualitatively and does not reflect the styles of majority of the domain. The focus of the classification is to explore what content-based navigation strategies are promising for each style of how-to video. Our keyword-based approach works best with domains located in the first quadrant. For videos that have lots of keywords and lots of actions, users gain confidence about being able to make this random access anchoring around the keywords.

However, other domains in the third quadrant that have less objects and less actions still remain unexplored with the challenge of extracting referenceable term from the transcript. For example, in the video tutorial for

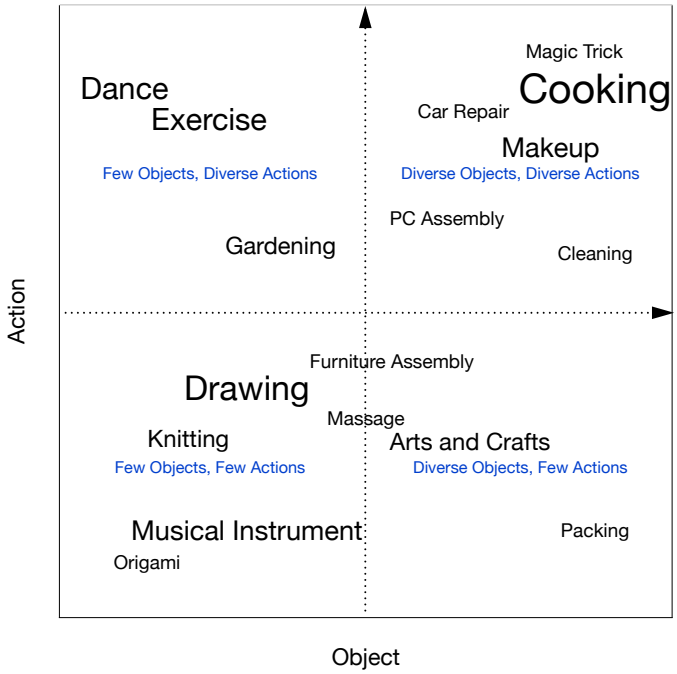


Figure 5.10: Analysis of most searched how-to video domains with respect to a variety of objects and actions (size logarithmically proportional to popularity)

origami, the words "corner, edge, fold, crease" are repeatedly used throughout the video, thus making the visual comparison inevitable.

Discussion throughout this paper limits the content-based search to matching the user query only with the transcript of the video not taking visual content into account. One possible line of future work is to investigate how adopting object detection and optical character recognition can extend the referenceable items in the absence of verbal descriptions. For example, when the person in the video shows an important item and refers to it as “this”, and has the keyword as graphic element on the scene for dramatic effect, our pipeline cannot detect the keyword despite its importance.

5.5.3 Leveraging Interaction at Scale

There are two dimensions in which we can consider scalability. One of them is anchoring around the tutorial video. Participants have suggested that they would be interested in seeing other viewers’ querying history. We also believe leveraging interaction traces like keywords and navigation patterns opens up opportunities for further advancing navigation interaction.

The other dimension is on the user. Once personal history accumulates, the system can infer what types of queries this specific user initiates, and possibly aim to understand the user struggle. Similar approaches have been explored in understanding usability issues of software [35].

5.5.4 Transferring to Voice Assistants

The user scenario that involves how-to videos by design include a visual display. We leveraged this fact, and were able to design effective visual pointers that guided users to navigate how-to videos.

However, we noticed one participant who had some familiarity with the task at hand successfully navigating without looking at the screen, but only verbally issuing commands. This was particularly interesting because it allows us to understand how to generalize the design of referencing strategies so it would be applicable to voice assistants without screens.

It would be a meaningful extension of this research to take a deeper look at what micro interactions are possible without the screen, and what the conditions are for successfully navigating a tutorial without visual display.

5.5.5 Is Mouse Interaction the Holy Grail?

The focus of our study was to design and implement voice-based content-based referencing for video interface. We did not compare RubySlippers against direct manipulation methods like mouse or keyboard as they correspond to different user scenarios and environments. However, whether effectively using temporal referencing and content-based referencing in voice user interfaces is as efficient as interacting directly with the timeline using the mouse is still an interesting question. If not, how do we design voice interactions so that it is? Perhaps can voice interfaces be more efficient than direct control if NLP and speech processing techniques advance? They are meaningful interaction problems, and we hope this research will catalyst future work in designing the next voice interfaces.

5.6 Conclusion

This paper presents RubySlippers, a voice-based navigation system for how-to videos. RubySlippers supports efficient content-based voice navigation through keyword-based queries. Our user study demonstrates that RubySlippers provides efficient, stress-free navigation for how-to videos in voice user interfaces.

Chapter 6. Workflow Graphs

6.1 Introduction

There are common situations in which many users of complex software perform the same task, such as designing a chair or table, bringing their unique set of skills and knowledge to bear on a set goal. For example, this occurs when multiple people perform the same tutorial, complete an assignment for a course, or work on sub-tasks that frequently occur in the context of a larger task, such as 3D modeling joints when designing furniture. It is also common for users to discuss and compare different methods of completing a single task in online communities for 3D modeling software (for an example of such discussion, see Figure 6.2). This raises an interesting possibility—what if the range of different methods for performing a task could be captured and represented as rich workflow recordings, as a way to help experienced users discover alternative methods and expand their workflow knowledge, or to assist novice users in learning advanced practices?

In this research, we investigate how multiple demonstrations of a fixed task can be captured and represented in a *workflow graph (W-graph)* (Figure 6.1). The idea is to automatically discover the different means of accomplishing a goal from the interaction traces of multiple users, and to encode these in a graph representation. The graph thus represents diverse understanding of the task, opening up a range of possible applications. For example, the graph could be used to provide targeted suggestions of segments of the task for which alternative methods exist, or to synthesize the most efficient means of completing the task from the many demonstrations encoded in the graph. It could also be used to synthesize and populate tutorials tailored to particular users, for example by only showing methods that use tools known to that user.

To investigate this approach, we instrumented *Tinkercad*[1], a 3D solid modeling application popular in the maker community, to gather screen recordings, command sequences, and changes to the CSG (constructive solid geometry) tree of the specific 3D model being built. The interaction traces for multiple users performing the same task are processed by an algorithm we developed, which combines them into a W-graph representing the collective actions of all users. Unlike past approaches to workflow modeling in this domain, which have focused on command sequence data (e.g., [111]), our approach additionally leverages the 3D model content being created by the user. This allows us to track the progress of the task in direct relation to changes in the content (i.e., the 3D model) to detect common stages of the task progression across multiple demonstrations. We use an autoencoder [86] to represent the 3D geometry information of each 3D model snapshot, which we found to be a robust and scalable method for detecting workflow-relevant changes in the geometry, as compared to metrics such as comparing CSG trees, 2D renders, and 3D meshes.

The result is a graph in which each directed edge from the starting node to a terminal node represents a potential workflow for completing the task, and multiple edges between any two states represent alternative approaches for performing that segment of the task. The collected command log data and screen recordings associated with the edges of the graph can be processed to define metrics on paths (such as average workflow duration or number of unique commands used), and displayed as demonstration content in interfaces.

The main contributions of this paper are:

[1] https://tinkercad.com

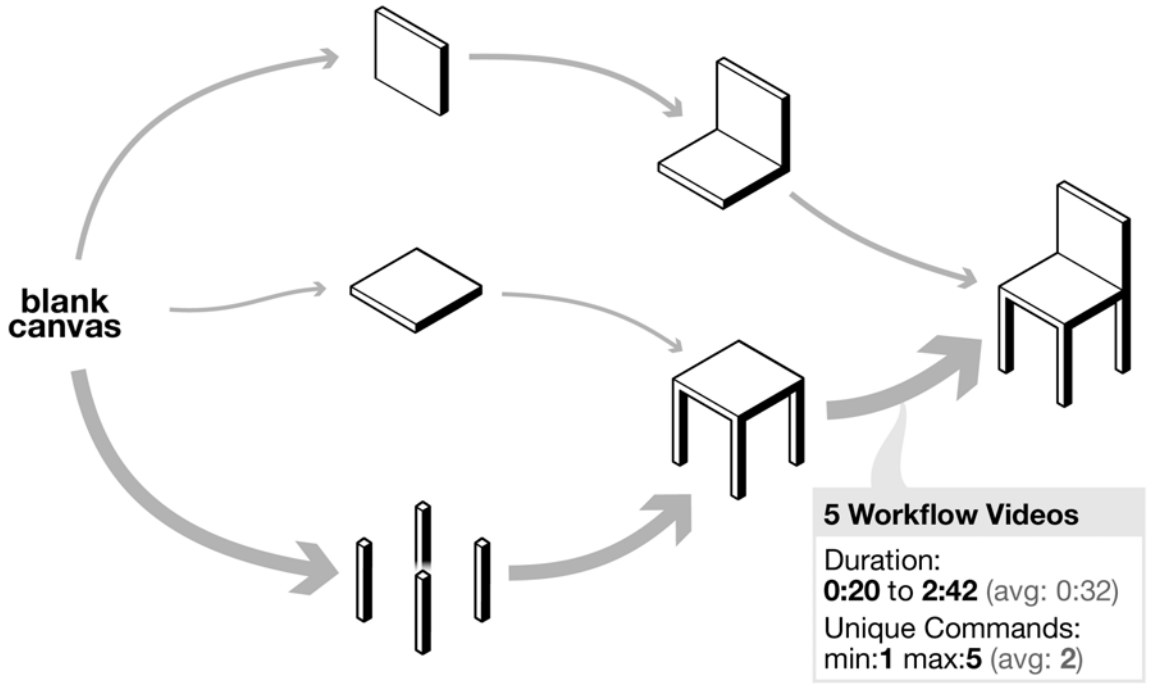


Figure 6.1: W-graphs encode multiple demonstrations of a fixed task, based on commonalities in the workflows employed by users. Nodes represent semantically similar states across demonstrations. Edges represent alternative workflows for sub-tasks. The width of edges represents the number of distinct workflows between two states.

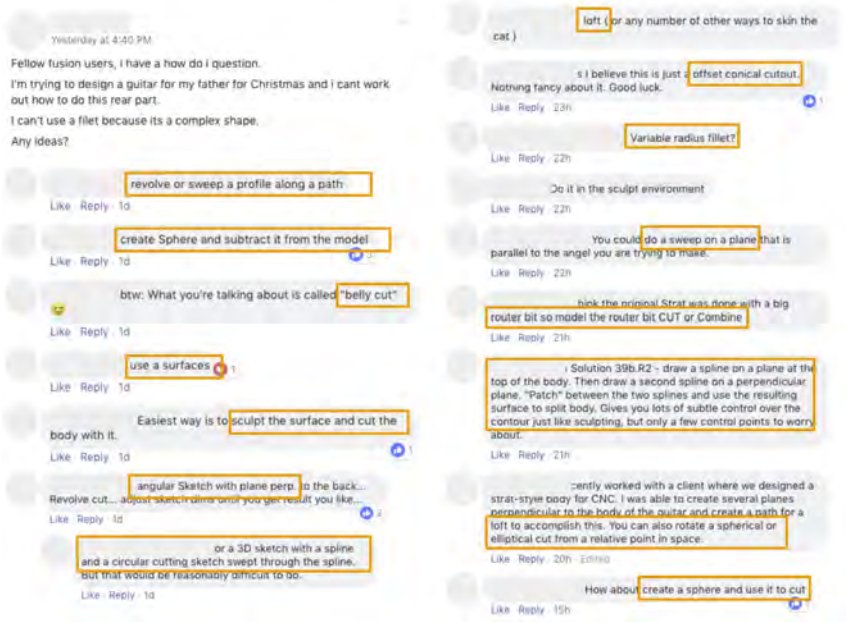


Figure 6.2: Fifteen distinct suggestions on how to perform a 3D modeling task – from the largest Fusion 360 user community on Facebook

The concept of W-graphs, which represent the semantic structure of a task, based on demonstrations from multiple users

A computational pipeline for constructing W-graph and a demonstration of the approach for sample tasks in Tinkercad

The description of possible applications enabled by W-graphs

We begin with a review of prior work, then describe the construction approach at a conceptual level. Next, we present workflow graphs constructed for two sample tasks performed by Tinkercad users, and discuss three applications enabled by —workflow feedback, on-demand task guidance, and instructor support. Finally, we present preliminary user feedback on a prototype of one of these applications, W-suggest, and conclude with a discussion of directions for future work.

6.2 Workflow Graphs

The key problem that we address is that designers and researchers currently lack scalable approaches for analyzing and supporting user workflows. To develop such an approach, we need techniques that can map higher level user intents (e.g., 3D modeling a mug), to strategy level workflows (e.g., modeling the handle before the body), and user actions (the specific sequence of actions involved).

We can broadly classify approaches for modeling user workflows derived from action sequences into bottom-up approaches and top-down approaches.

Bottom-up approaches record users' action sequences, and then attempt to infer the user's intent at a post-processing stage using unsupervised modeling techniques such as semantic segmentation, clustering, or topic modeling [16, 4]. A disadvantage of this approach is that the results can be difficult to present to users, because the results of unsupervised modeling techniques are not human-readable labels. Meaningful labels could conceivably be added to the resulting clusters (e.g., using crowdsourcing techniques [22, 101, 55]), but this is a non-trivial problem under active research.

An alternative is a top-down approach, in which a small number of domain experts break down a task into meaningful units (e.g., subgoals [15]), and then users or crowdworkers use these pre-created units as labels for their own command log data, or that of other users. This approach also comes with disadvantages—users must perform the labeling, their interpretation of pre-defined labels can differ, and the overall breakdown of the task depends on the judgement of a few domain experts, limiting the scalability of the approach.

Then, how can we develop an approach for organizing users' collective interaction data into a meaningful structure while maintaining the scalability of naively recording user action sequences without interrupting them to acquire any labels?

To investigate this possibility, we developed Workflow graphs (W-graphs), which synthesize many demonstrations of a fixed task (i.e., re-creating the same 3D model) such that the commonalities and differences between the approaches taken by users are encoded in the graph. To ensure the technique can scale, the goal is to automate the construction process, using recordings of demonstrations of the task as input (which may include screen recordings, command log data, content snapshots, etc.).

Formally, a W-graph is a directed graph $G = (V, A)$ which consists of the following components:

6.2.1 Graph Vertices

$$V = \{v_i; 1 \leq i \leq N\}$$

The vertices of the graph represent semantically-meaningful states in the demonstrations, such as a sub-goal of the task. These states can be thought of as sub-goals in the workflow—ideally, we want them to capture the points where a user has completed a given sub-task, and has yet to start the next sub-task. Detecting these states automatically from unlabeled demonstrations is a challenge, but the idea is to leverage the demonstrations of multiple users to discover common states that occur across their respective methods for completing the task. If a new demonstration is completely different from those already represented in the graph, it might not share any nodes with those already in the graph, apart from the start and final nodes, which are shared by all demonstrations.

Note that the appropriate criteria for judging which states from multiple demonstrations are semantically-similar is ill-defined, and dependent on the intended application of the W-graph. For example, one criterion could be used to construct a W-graph that indicates coarse differences between approaches for completing the task, while a more strict criterion for similarity could create a more complex graph, which reveals finer differences between similar approaches. As we discuss in the next section, our algorithm allows the threshold for the similarity to be tuned based on the intended application.

6.2.2 Graph Edges

$$A = \{(v_i, v_j, d_k, E_{i,j}); v_i, v_j \in V\}$$

$$E_{i,j,k} = \{event_1, event_2, event_3, \ldots\}$$

The directed edges of the graph represent workflows used by a user to move between semantically-similar states. There may be multiple directed edges between a given pair of states, if multiple demonstrations d_k include a segment from state v_i to v_j.

Each directed edge is associated with a set of events $E_{i,j,k}$ which include the timestamped interaction trace of events in demonstration d_k performed in the segment between state v_i and v_j. This trace of events could includes timestamped command invocations, 3D model snapshots, or any other timestamped data that was gathered from the recorded demonstrations.

6.2.3 Interaction Data

The interaction trace data associated with edges enables a great deal of flexibility in how the is used. For example, this data could be used to retrieve snippets of screen recordings of the demonstrations associated with the segment of the task between two states, or it could be used to define metrics on the different workflows used for that segment of the task (e.g., the number of unique commands used, or the average time it takes to perform the workflow). As another example, analyzing the interaction traces along many different paths between states can reveal the average time for sub-tasks or the variance across users. Later in the paper, we present some example applications of to illustrate the full flexibility of this data representation.

6.3 Pipeline for Constructing

In this section we describe the computational pipeline we have developed for constructing . We start by discussing our instrumentation of Tinkercad and the data set we collected, then present the multi-step pipeline for processing the data and the similarity metric for identifying equivalent-intermediate states. The choice of a method for identifying equivalent-intermediate states is a key aspect of the pipeline, and we experimented with several alternative methods.

6.3.1 Tinkercad Data Collection

We instrumented a customized version of Tinkercad to record timestamped command invocations and snapshots of the 3D model the user is working on after each command is executed (represented as a constructive solid geometry (CSG) tree with unique IDs for each object, to enable the association of model parts across multiple snapshots). To capture the instrumentation data, participants were asked to install Autodesk Screencast[2], a screen recording application that can associate command metadata with the timeline of recorded video data. Collectively, this allowed us to gather timestamp-aligned command invocation data, 3D model snapshots, and screen recordings of participants performing 3D modeling tasks. An example of a user-recorded screencast video can be seen in Figure 6.3.

Using this approach, we collected user demonstrations for two tasks—modeling a mug and modeling a standing desk (Figure 6.4). These tasks were selected because they could be completed in under 30 minutes, and represent different levels of complexity. The mug task is relatively simple, requiring fewer operations and primitives, while the desk task can be complex and time consuming if the user does not have knowledge of particular Tinkercad tools, such as the Align and Ruler. The Desk model also requires approximately twice as many primitives as the Mug model.

We recruited participants through UserTesting.com and an email to an internal mailing list at a large software company. 14 participants were recruited for the Mug task, and 11 participants were recruited for the Desk task, but we excluded participants who did not follow the instructions, or failed to upload their recordings in the final step. After applying this criteria, we had 8 participants for the mug task (6 male, 2 female, ages 27–48), and 6 participants for the standing desk task (5 male, 1 female, ages 21–43).

The result of data collection procedure were 8 demonstrations for the Mug task, which took 26m:24s on average (SD=10m:46s) and consisted of an average of 142 command invocations (SD=101); and 6 demonstrations for the Desk task, which took 23m:23s on average (SD=8m:20s) and consisted of an average of 223 command invocations (SD=107).

6.3.2 Workflow to Graph Construction

The construction pipeline consists of three steps: preprocessing, collapsing node sequences, and sequence merging.

[2]https://knowledge.autodesk.com/community/screencast

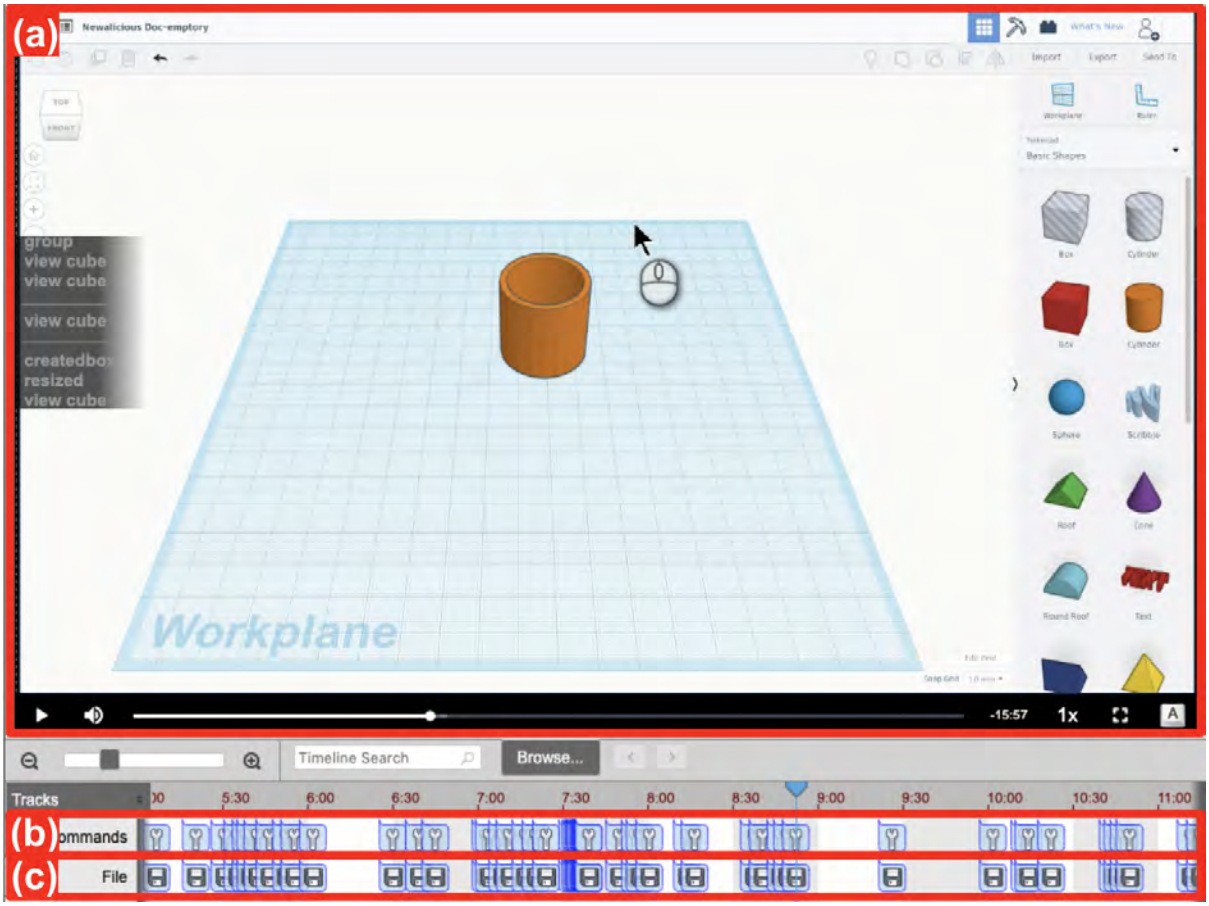


Figure 6.3: Screencast of a user demonstration, consisting of the (a) screen recording, (b) command sequences, and (c) 3D model snapshots

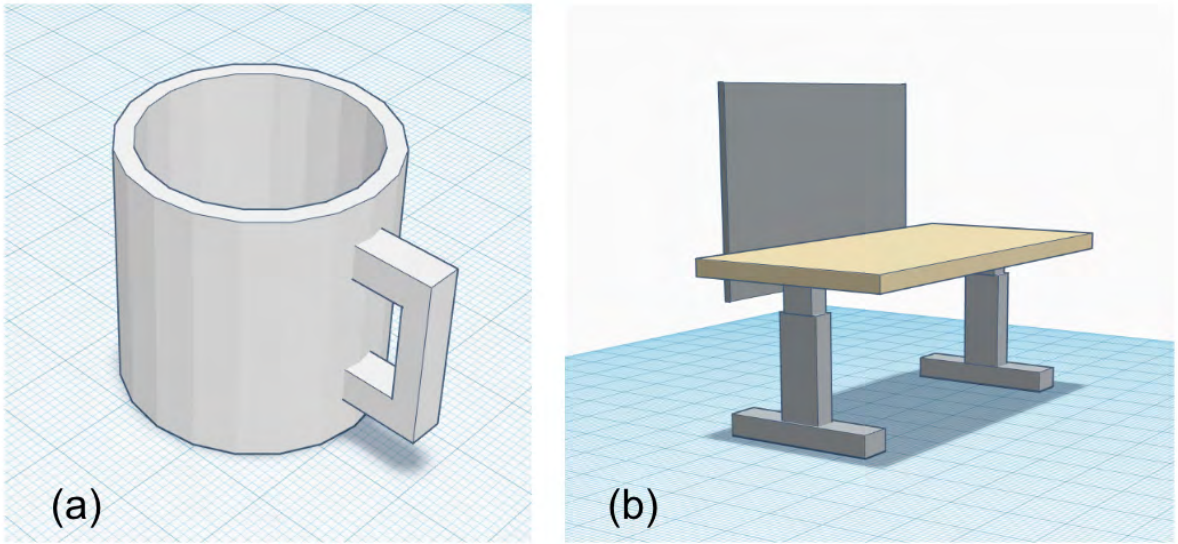


Figure 6.4: Models used for data collection – (a) Mug, (b) Desk

Step 1. Preprocessing

To start, we collapse repeated or redundant commands in the sequence of events (both keystroke and click-stream data) for each demonstration. For example, multiple invocations of "arrow key presses" for moving an

object are merged into one “object moved with keyboard” and multiple invocations of “panning viewpoint” are merged into “panning”.

Next, the sequence of events for each user is considered as a set of nodes (one node per event), with directed edges connecting each event in timestamped sequence (Figure 6.5a). The 3D model snapshot for each event is associated with the corresponding node, and the event data (including timestamped command invocations) is associated with the incoming edge to that node. Since each demonstration starts from a blank document and finishes with the completed 3D model, we add a START node with directed edges to the first node in each demonstration, and we merge the final nodes of each demonstration into an END node. At this point, each demonstration represents a distinct directed path from the START node to the END node (Figure 6.5b).

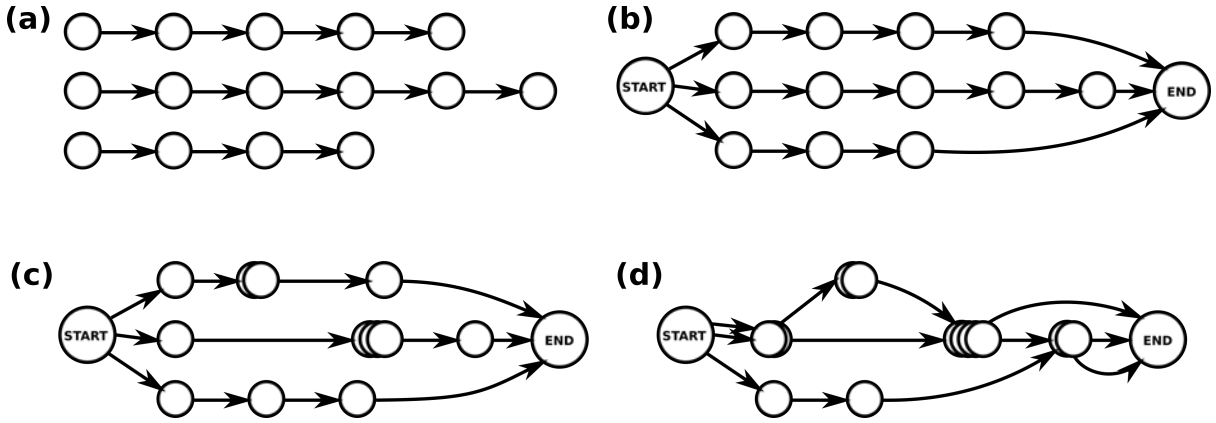


Figure 6.5: Illustration of how sequences get compressed and merged into a

Step 2. Collapsing Node Sequences

Next, the pipeline merges sequences of nodes with similar geometry along each path from START to END, by clustering the snapshots of 3D model geometry associated with the nodes along each path (Figure 6.5c). The metric we use for 3D model similarity is discussed at the end of this section. To identify sequences with similar geometry, we first apply the DBSCAN [32] algorithm to cluster the 3D model snapshots associated with each path. We then merge contiguous subsequences of nodes that were assigned to the same cluster, keeping the 3D model snapshot of the final state in the subsequence as the representation of that node. We selected DBSCAN because it does not require a pre-defined number of clusters, as in alternative clustering algorithms such as K-Means. The hyperparameters of DBSCAN are tuned using the K-Nearest Neighborhood distance method, which is a standard practice for this algorithm [12, 94, 9].

Step 3. Sequence Merging

Finally, the pipeline detects “equivalent-intermediate” nodes across the paths representing multiple demonstrations (Figure 6.5d). To do this, we compute the 3D model similarity metric for all pairs of nodes that are not associated with the same demonstration (i.e., we only consider pairs of nodes from different demonstrations). We then merge all nodes with a similarity value below a threshold ϵ that we manually tuned. In our experience, varying ϵ can yield graphs that capture more or less granularity in variations in the task, and it would be interesting to consider an interactive system in which users can select a granularity that is suited to their use of the .

At this point, the construction is complete. As at the start of the pipeline, the directed edges from START to END collectively include all the events from the original demonstrations, but now certain edges contain multiple events (because the nodes between them have been collapsed), and some nodes are shared between multiple demonstrations.

6.3.3 Metrics for Detecting "Equivalent-intermediate States"

The most crucial part of the pipeline is determining the "similarity" between 3D model snapshots, as this is used to merge sequences of events in demonstrations, and to detect shared states across multiple demonstrations. We experimented with four different methods of computing similarity between 3D model snapshots, which we discuss below.

Comparing CSG trees

3D model snapshots are represented as CSG trees by Tinkercad, which consist of geometric primitives (e.g., cubes, cylinders, cones), combined together using Boolean operations (e.g., union, intersection, difference) in a hierarchical structure. A naive method of quantifying the difference between two snapshots would be to compare their respective trees directly, for example by trying to associate corresponding nodes, and then comparing the primitives or other characteristics of the tree. However, we quickly rejected this method because different procedures for modeling the same geometry can produce significantly different CSG trees. This makes the naive CSG comparison a poor method of judging similarity, where we specifically want to identify states where a similar end-result was reached through distinct methods.

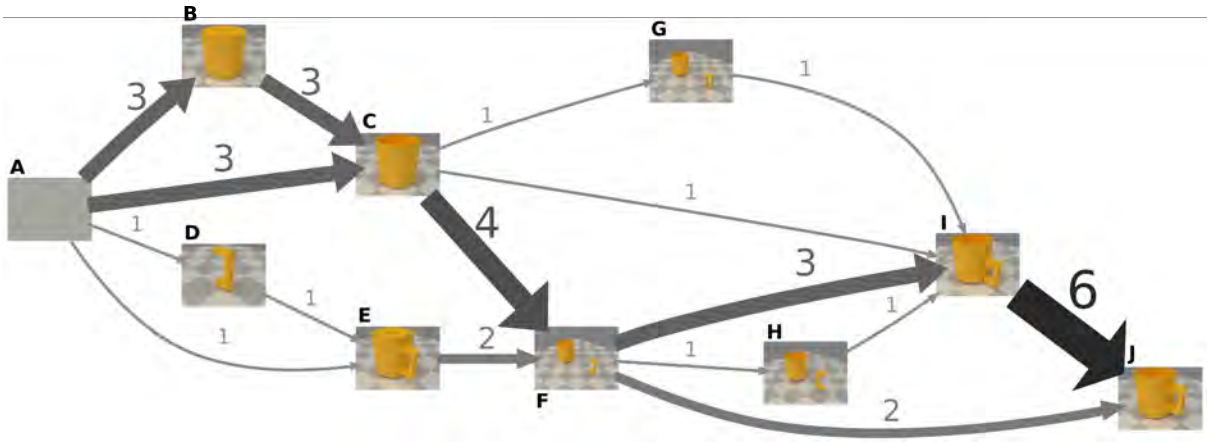


Figure 6.6: W-graph for the mug task. Edge labels indicate the number of demonstrations for each path. For nodes with multiple demonstrations, a rendering of the 3D model snapshot is shown for one of the demonstrations. A high-res version of this image is included in supplementary materials.

Comparing 2D Images of Rendered Geometry

Inspired by prior work that has used visual summaries of code structure to understand the progress of students on programming problems [115], we next explored how visual renderings of the models could be used to facilitate comparison. We rendered the geometry of each 3D model snapshot from 20 different angles, and then compared the resulting images for pairs of models to quantify their difference. The appeal of this approach is that the method

used to arrive at a model does not matter, so long as the resulting models look the same. However, we ultimately rejected this approach due to challenges with setting an appropriate threshold for judging two models as similar based on pixel differences between their renders.

Comparing 3D Meshes

Next, we experimented with using the Hausdorff distance [6], a commonly used mesh comparison metric, to compare the 3D meshes of pairs of 3D model snapshots. As with the comparison of rendered images, this method required extensive trial and error to set an appropriate threshold. However, the biggest drawback of this method was that the distances produced by the metric are in absolute terms, with the result that conceptually minor changes to a 3D model, such as adding a cube to the scene, can lead to huge changes in the distance metric. Ideally we would like to capture how "semantically" meaningful changes are, which is not always reflected in how much of the resulting mesh has been altered.

Latent Space Embedding using Autoencoders

The final method we tried was to use an autoencoder to translate 3D point cloud data for each 3D model snapshot into a 512-dimensional vector. Autoencoders learn compact representations of input data by learning to encode a training set of data to a latent space of smaller dimensions, from which it can decode to the original data. We trained a latent model with a variation of PointNet [86] for encoding 3D point clouds to vectors, and PointSet Generation Network [33] for decoding vectors back to point clouds. The model was trained using the ShapeNet [119] dataset, which consists of 55 common object categories with about 51,300 unique 3D models. By using an additional clustering loss function [116], the resulting distributed representation captures the characteristics that matter for clustering tasks. One of the limitations of PointNet autoencoders is that current techniques cannot perform rotational-invariant comparisons of geometries. However, this fits nicely with our purpose, because rotating geometry does not affect semantic similarity for the 3D modeling tasks we are targeting.

Once trained, we can use the autoencoder to produce a 512-dimensional vector for each 3D model snapshot, and compare these using cosine distance to quantify the similarity between models. Overall, we found this to be the most effective method. Because it works using 3D point cloud data, it is not sensitive to how a model was produced, just its final geometry. Moreover, it required less tuning than comparing 2D images of rendered geometry or comparing 3D meshes, and in our experiments appeared to be more sensitive to semantically-meaningful changes to models.

6.3.4 Results

As a preliminary evaluation of the pipeline, we examined the graphs constructed for the mug and standing desk tasks. The for the mug task is shown in Figure 6.6. From the graph, a few things can be observed. First, the high-level method followed by most users was to first construct the body of the mug (as seen in paths A-B-C, and A-C), and then build and add the handle. Examining the screen recordings, all three users on path A-B-C created the body by first adding a solid cylinder and then adding a cylindrical "hole" object[3] to hollow out the center of the solid cylinder (see Figure 6.7a). Two of the three users on path A-C followed a slightly different method, creating

[3]Tinkercad shapes can be set as solid or as *holes*, which function like other shapes but cut out their volume when grouped with solid objects.

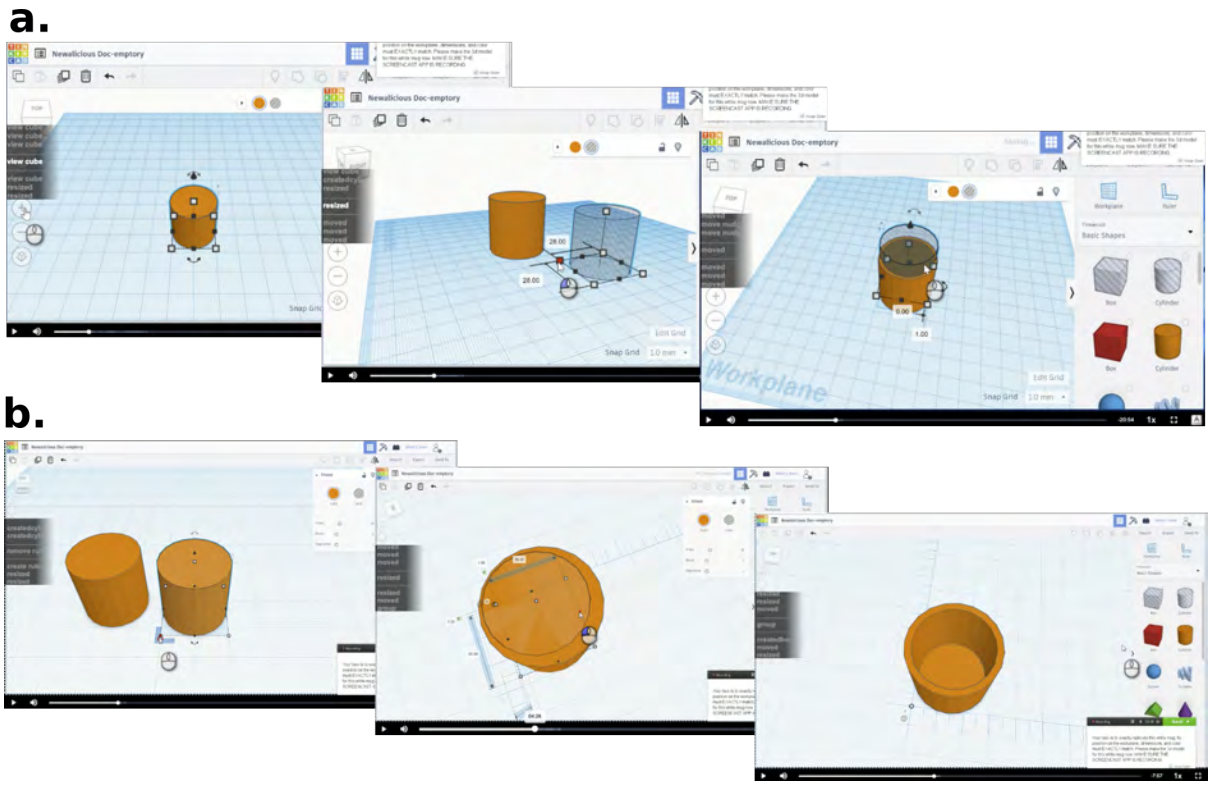


Figure 6.7: Two distinct methods of creating the mug body: (a) Create a solid cylinder, create a cylindrical hole, and group them; (b) Create two solid cylinders, position them correctly, then convert one into a hole.

two solid cylinders first, and then converting one of them into a hole object (Figure 6.7b). It is encouraging that the pipeline was able to capture these two distinct methods.

The remaining user on path A-C created a hole cylinder first, but ultimately deleted it and started again, following the same procedure as the users on path A-B-C. This highlights an interesting challenge in building , which is how to handle backtracking or experimentation behavior (using commands such as Undo and Erase). We revisit this in the Discussion section at the end of the paper.

The users on paths A-D-E-F and A-E-F followed a different approach from those discussed above. Both of these users started by creating a cylinder (as a hole in the case of A-D-E-F, and as a solid in the case of A-E-F), then built the handle, and finally cut out the center of the mug's body. The A-D-E-F user built the handle through the use of a solid box and a hole box (Figure 6.8a), but the A-E-F user used a creative method—creating a primitive in the shape of a letter 'B', then cutting out part of it to create the handle (Figure 6.8b). Again, it is encouraging that the pipeline was able to separate these distinct methods.

For the modeling of the handle, nodes F, G, and H capture the behavior of building the handle apart from the body of the mug, and then attaching it in states I and J. The E-F transition seems strange in Figure 6.6, but reviewing the screen recording, the user moved the handle away from the mug before cutting the hole in the body, perhaps to create some space to work.

Overall, the pipeline appears to be effective in capturing the variety of methods used to create the body of the mug, and the edges of the graph captured a few distinct methods for creating the handle. An interesting observation is that the node identification algorithm did not capture any sub-steps involved in creating the handle. One possibility is that the methods used by different users were distinct enough that they did not have any

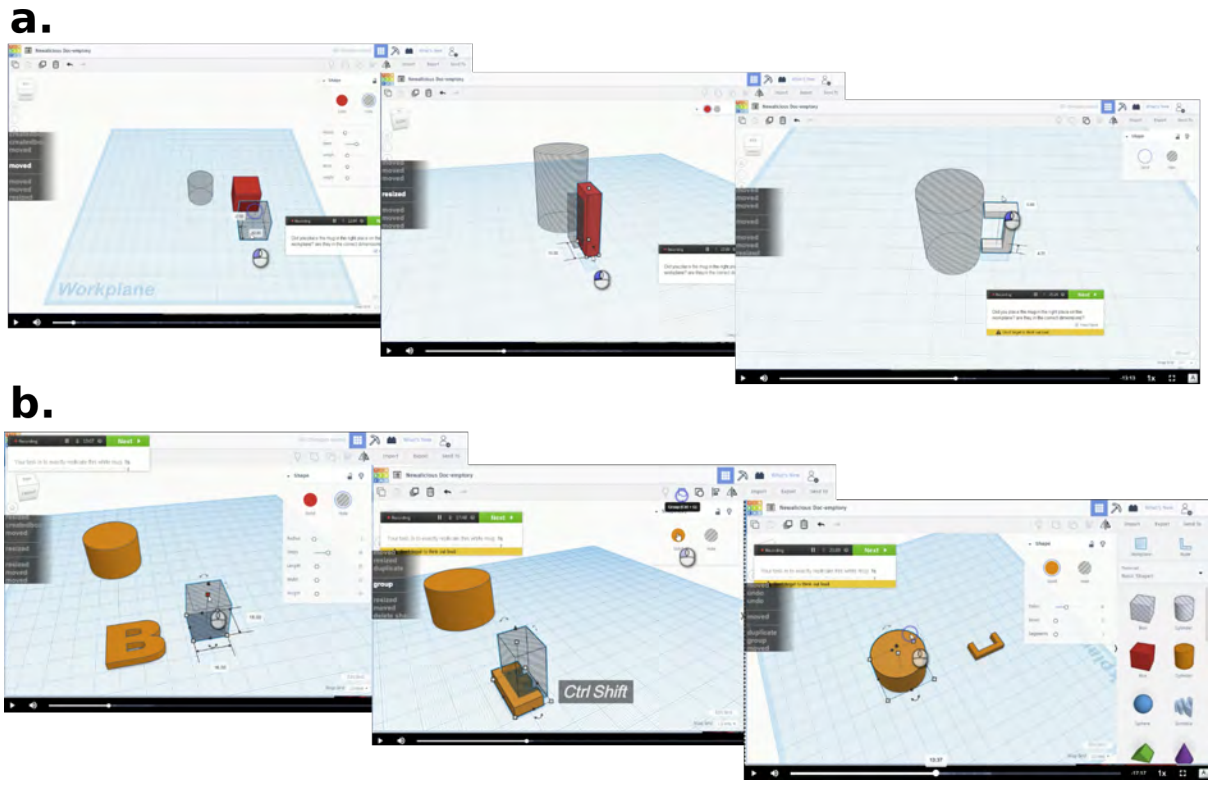


Figure 6.8: Two methods of creating the handle: (a) Combine a solid box and a box-shaped hole; (b) Cut a letter 'B' shape into the handle using several box-shaped holes.

Figure 6.9: W-graph for the standing desk task. Edge labels indicate the number of demonstrations for each path. For nodes with multiple demonstrations, a rendering of the 3D model snapshot is shown for one of the demonstrations. A high-res version of this image is included in supplementary materials.

equivalent-intermediate states until the handle was complete. Another possibility is that the autoencoder is not good at identifying similar states for models that are partially constructed (being trained on ShapeNet, which consists of complete models). The above having been said, this is not necessarily a problem as the edges do capture multiple methods of constructing the handle.

The for the standing desk task is shown in Figure 6.9. The graph is more complex than that for the mug task, reflecting the added complexity of creating the standing desk, but we do observe similarities in how the graph captures the task. In particular, we can see paths that reflect the different orders in which users created the three main parts of the desk (the top, the legs, and the privacy screen).

We also notice some early nodes with box shapes, which later diverge and become a desk top in some demonstrations, and legs in another. These nodes that represent a common geometric history for different final shapes are interesting, because they represent situations where the algorithm may correctly merge similar geometry, but doing so works counter to the goal of identifying workflows for completing sub-goals of the task, effectively

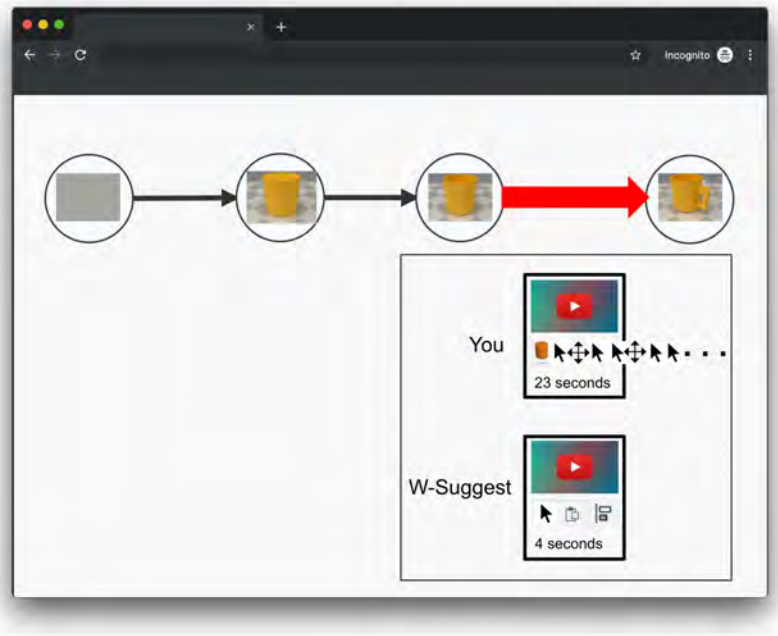


Figure 6.10: W-Suggest – A workflow suggestion interface mockup

breaking them up into several edges. A possible way to address this would be to modify the pipeline so it takes into account the eventual final placement of each primitive at the end of the task, or several edges forward, in determining which nodes to merge.

6.4 Potential Applications of W-Graphs

This section presents three novel applications that are made possible by : 1) W-Suggest, a workflow suggestion interface, 2) W-Guide, an on-demand 3D modeling help interface, and 3) W-Instruct, an instructor dashboard for analyzing workflows.

6.4.1 W-Suggest: Workflow Suggestion Interface

By representing the structure of how to perform a task, can serve as a back-end for applications that suggest alternate workflows to users.

To use the W-Suggest system(Figure 6.10), the user first records themselves performing a 3D modeling task, similar to the procedure performed by participants in the previous section. However, instead of integrating this new workflow recording into the W-graph, the system compares the workflow to the existing graph and suggests alternate workflows for portions of the task.

W-Suggest uses the following algorithm to make its suggestions. First, it performs Steps 1 and 2 of the W-graph construction pipeline on the user's recording of the task (i.e., preprocessing the events, and collapsing node sequences with similar geometry). Next, the 512-dimensional embedding vector for each remaining 3D model snapshot is computed using the same autoencoder used for the construction pipeline. The vectors for each of these nodes are then compared to those of the nodes along the shortest path from START to END (as measured by total command invocations) to detect matches using the same threshold ϵ used for graph construction. Finally,

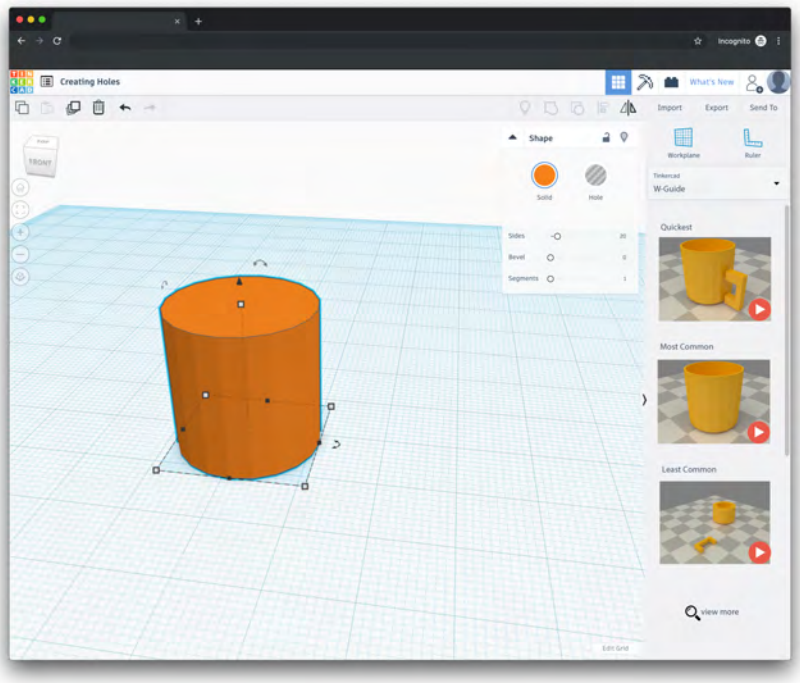


Figure 6.11: W-Guide – An on-demand task guidance interface mockup.

for each pair of matched nodes (one from the user, one from the shortest path in the), the edge originating at the user's node and the edge originating at the node are compared based on command invocations. Based on all of these comparisons, the algorithm selects the pair for which there is the biggest difference in command invocations between the user's demonstration and the demonstration from the W-graph. In effect, the idea is to identify segments of the user's task for which the W-graph includes a method that uses much fewer command invocations, which can then be suggested to the user.

6.4.2 W-Guide: On-Demand Task Guidance Interface

W-graphs could also serve as a back-end for a *W-Guide* interface that presents contextually appropriate video content to users on-demand as they work in an application, extending approaches taken by systems such as Ambient Help [70] and Pause-and-Play [84] with peer demonstrations.

While working on a 3D modeling task in Tinkercad, the user could invoke W-Guide to see possible next steps displayed in a panel to the right of the editor (Figure 6.11). These videos are populated based on the strategies captured from other users and stored in the W-graph. Specifically, the panel recommends video demonstrations from other users matched to the current user's state, and proceeds to the next "equivalent-intermediate" state (i.e., one edge forward in the graph). Using a similar approach to W-Suggest, these can be provided with meaningful labels (e.g., "Shortest workflow", "Most popular workflow", etc.).

W-Guide could use the identical algorithm as W-Suggest to construct a W-Graph and populate its suggestions. The only difference is that it would attempt to match the user's current incomplete workflow to the graph. This is achievable because the ϵ threshold for collapsing node sequences is flexible, allowing W-Guide to construct a W-Graph from any point in current user's workflow and populate demonstrations for next steps.

An exciting possibility that becomes possible with W-Guide is that the system could dynamically elicit ad-

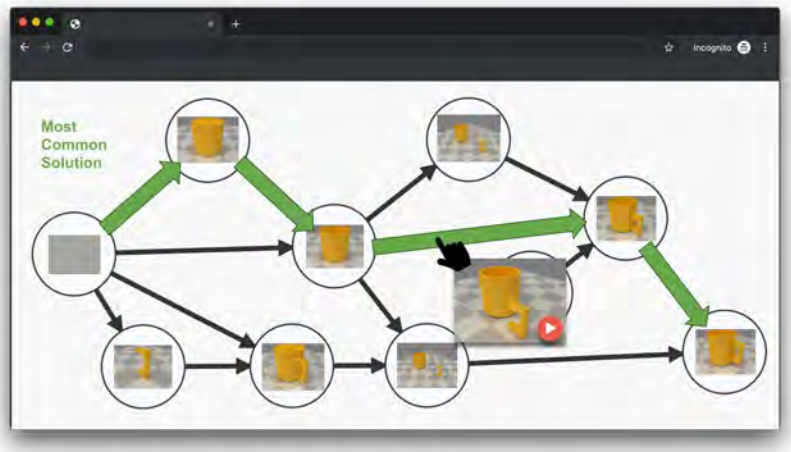


Figure 6.12: W-Instruct – An instructor tool mockup.

ditional demonstrations from users in a targeted way (e.g., by popping up a message asking them to provide different demonstrations than those pre-populated in the panel). This could allow the system to take an active role in fleshing out a W-graph with diverse samples of methods.

6.4.3 W-Instruct: Instructor Tool

Finally, we envision the *W-Instruct* system in which W-graphs become a flexible and scalable tool for instructors to provide feedback to students, assess their work, and generate tutorials or other instructional materials on performing 3D modeling tasks.

W-Instruct (Figure 6.12) supports instructors in understanding the different methods used by their students to complete a task—by examining the graph, an instructor can see the approaches taken by students, rather than simply the final artifacts they produce. The grouping of multiple students' workflows could also be used as a means to provide feedback to a large number of learners at scale (e.g., in a MOOC setting). Also, the instructor could quickly identify shortcuts, crucial parts of the workflow to emphasize, or common mistakes by browsing the W-Graph. As shown in Figure 6.12, edges can be highlighted to show the most common solutions, and the video demonstration corresponding to an edge can be viewed by hovering over a node in the graph.

Along similar lines to W-Instruct, we see potential for to support the generation of tutorials and other learning content, building on past work exploring approaches for generating tutorials by demonstration [39, 21]. For example, synthetic demonstrations of workflows could potentially be produced that combine the best segments of multiple demonstrations in the , creating a demonstration that is more personalized to the current user than any individual demonstration.

6.5 User Feedback on W-Suggest

While the main focus of this work is on the computational approach for constructing W-graphs, we implemented the W-Suggest application as a preliminary demonstration of the feasibility of building applications on top of a constructed W-graph (Figure 6.13). The W-Suggest interface consists of a simplified representation of the user's workflow, with edges highlighted to indicate a part of the task for which the system is suggesting an

improved workflow. Below this are two embedded video players, one showing the screen recording of the user's workflow for that part of the task, and the other showing a suggested workflow drawn from other users in the graph. Below this are some metrics on the two workflows, including duration, the distribution of commands used, and the specific sequences of commands used.

To gain some feedback on the prototype, we recruited 4 volunteers to perform one of the two tasks from the previous section (two for the mug task, two for the standing desk task) and presented them with their W-Suggest interface. We asked them to watch the two videos—one showing their workflow, the other showing the suggested workflow—and then asked a few short questions about the interface. Specifically, we asked if they felt it was useful to view the alternate demonstration, and why or why not they felt that way. We also asked them their thoughts on the general utility of this type of workflow suggestion system, and what aspects of workflows they would like suggestions on for software they frequently use.

Due to the small number of participants for these feedback sessions, they are best considered as providing preliminary feedback, and certainly not a rigorous evaluation. That being said, the feedback from participants was quite positive, with all participants agreeing it would be valuable to see alternative workflows. In particular, participants mentioned that it would be valuable to see common workflows, the fastest workflow, and workflows used by experts.

Two participants mentioned that they learned something new about how to use Tinkercad from watching the alternate video, as in the following quote by P2 after seeing a use of the Ruler tool to align objects: *Oh, you can adjust the things there [with the Ruler] that's useful. Oh, there's like an alignment thing, that seems really easy.*

Likewise, P4 observed a use of the Workplane tool that he found valuable: *It's assigning relative positions with it [the Workplane and Ruler]—I wanted to do something like that.*

All participants agreed that efficiency is an important criterion when recommending alternative workflows. However, P1 and P2 noted that the best method to use in feature-rich software, or other domains such as programming, can often depend on contextual factors. In particular, P1 noted that they might prepare a 3D model differently if it is intended to be 3D printed. This suggests that additional meta-data on the users or the intended purpose for creating a model could be useful for making workflow recommendations.

6.6 Discussion, Limitation and Future Work

Overall, the produced for the mug and standing desk tasks are encouraging, and suggest that our pipeline is effective at capturing different high-level methods for modeling 3D objects. Testing the pipeline on these sample tasks also revealed a number of potential directions for improving the approach, including modeling backtracking behavior in demonstrations, and accounting for sub-tasks with common intermediate states. Finally, our user feedback sessions for W-Suggest showed enthusiasm for applications built on W-graphs, and revealed insights into criteria for what makes a good demonstration, including the importance of contextual factors.

In this section we revisit the potential of modeling backtracking and experimentation, discuss the question of how many demonstrations are needed to build a useful W-graph graph, and suggest further refinements of the graph construction method. We then discuss how our approach could be generalized to building models of similar tasks.

Figure 6.13: W-Suggest – The implemented interface.

6.6.1 Backtracking and Experimentation

In our current approach, Undo and Erase are treated the same as other commands. In some situations this may be appropriate, but at other times these commands may be used to backtrack, to recover from mistakes, or to try other workflows, and past work has shown their occurrence may indicate usability challenges [2]. It would be interesting to investigate whether these practices for using Undo and Erase could be detected and represented in a . This could take the form of edges that go back to previous states, creating directed cycles or self-loops in the graph. Applications built on top of a W-graph could also use the number of Undos as a metric for ranking paths through the graph (e.g., to identify instances of exploratory behavior), or as a filtering metric to cull the graph of such backtracking behavior.

6.6.2 Branching Factors and Graph Saturation

A nice feature of W-graphs is that they can be built with only a few demonstrations. As the number of demonstrations grows, the graph can more fully capture the space of potential workflows for the task. However, it is likely that the graph will eventually reach a point at which it is *saturated*, beyond which additional workflows will contribute a diminishing number of additional methods. The number of demonstrations needed to reach saturation will likely vary task by task, with more complex tasks requiring more demonstrations than simpler tasks. Examining how the sum of the branching factor for all nodes in the tree changes with each added demonstration may give an indication of when the graph has reached saturation, as the number of branches is likely to stop growing once new methods are no longer being added.

6.6.3 Scalability

In one sense, the W-graph approach is scalable by design, as it relies on computational comparisons of 3D models rather than human interventions such as expert labeling or crowdsourcing. However, more work is needed to understand how the structure of W-graphs produced by our pipeline change as the number of demonstrations in a graph grows. In particular, there is the question of how the parameters for identifying similar intermediate states may need to change in response to a growing number of workflows, in order to produce graphs at the right granularity for a given application, and other issues that may come up when processing many demonstrations. On the application end, metrics could be developed to identify less-used but valuable traces contained in a graph with many demonstrations.

6.6.4 Robustness Against Different Workflow Orders

A potential limitation of our current approach is that it preserves the global order of sub-tasks, including those that could be performed in an arbitrary order (e.g., a user could start by modeling the legs or the top of a table), and this could prevent it from grouping some variations of sub-tasks together if a given sub-task is performed first by some users, and later by others. Preserving the global order of sub-tasks has some advantages, in that it reveals how users commonly sequence the sub-tasks that make up the overall task, and it can also reveal cases where sub-tasks benefit from being ordered in a certain way, as may occur when objects built as part of a preceding sub-task are used to help with positioning or building objects in a subsequent sub-task. However, it would be interesting to look at approaches that post-process a W-graph to identify edges across the graph where the same sub-task is

being performed (e.g., by looking for edges where similar changes to geometry are made, ignoring geometry that isn't changing) to address this limitation and gain insights into sub-task order in the graph.

6.6.5 Extension to Similar Tasks and Sub-Tasks

Another interesting direction for future work is to consider how the W-graph approach could be extended to scenarios where the demonstrations used to produce the graph are not of the exact same task, but instead represent workflows for a class of similar tasks (e.g., modeling chairs). We believe the autoencoder approach we have adopted could be valuable for this, as it is less sensitive to variations in the model, and potentially able to capture semantic similarities between models of different objects within a class, but more research is required. Sub-goal labels provided by users or learners could be valuable here, building on approaches that have been used for how-to videos [52] and math problems [114]. Given a user's explanation of their process or different stages in the task, the graph construction algorithm would have access to natural language descriptions in addition to interaction traces and content snapshots, which could be used to group workflows across distinct but related tasks.

Beyond refining our algorithms to work with similar tasks, it would be interesting to investigate how a large corpus of demonstrations could be mined to identify semantically-similar sub-tasks (which could be then turned into). *Multi-W-graphs* could conceivably be developed that link together the nodes and edges of individual W-graphs, to represent similarities and relationships between the workflows used for different tasks. For example, nodes representing the legs of a desk, chair, or television stand could be linked across their respective graphs, and edges that represent workflows for creating certain effects (e.g., a particular curvature or geometry) could be linked as well. In the limit, one could imagine a set of linked graphs that collectively encode all the tasks commonly performed in a domain, and feed many downstream applications for workflow recommendation and improvement.

6.6.6 Generalizing to Other Software and Domains

Though we demonstrated our approach for 3D modeling software, the construction approach would be straightforward to extend to other software applications and domains. For many domains, such as 2D graphics or textual media, the technique could be generalized by simply substituting in an appropriate feature extraction mechanism for that domain. More challenging would be extending the approach to apply across a variety of software applications, perhaps by different software developers, where instrumentation to gather commands and content is not easy. To approach this, we could imagine using the screen recording data for the content, and accessibility APIs to gather the actions performed by users (an approach used in recent work [37]). Beyond fully-automated approaches, learnersourcing approaches [52] could be used to elicit sub-goals that have particular pedagogical value, and these peer-generated sub-goals could be turned into feedback for other learners in the system, using similar methods to those explored in other applications [47].

6.7 Conclusion

This work has contributed a conceptual approach for representing the different means by which a fixed goal can be achieved in feature rich software, based on recordings of user demonstrations, and has demonstrated a scalable pipeline for constructing such a representation for 3D modeling software. It has also presented a range of applications that could leverage this representation to support users in improving their skill sets over time. Overall,

we see this work as a first step toward enabling a new generation of help and learning systems for feature-rich software, powered by data-driven models of tasks and workflows.

Chapter 7. Discussion

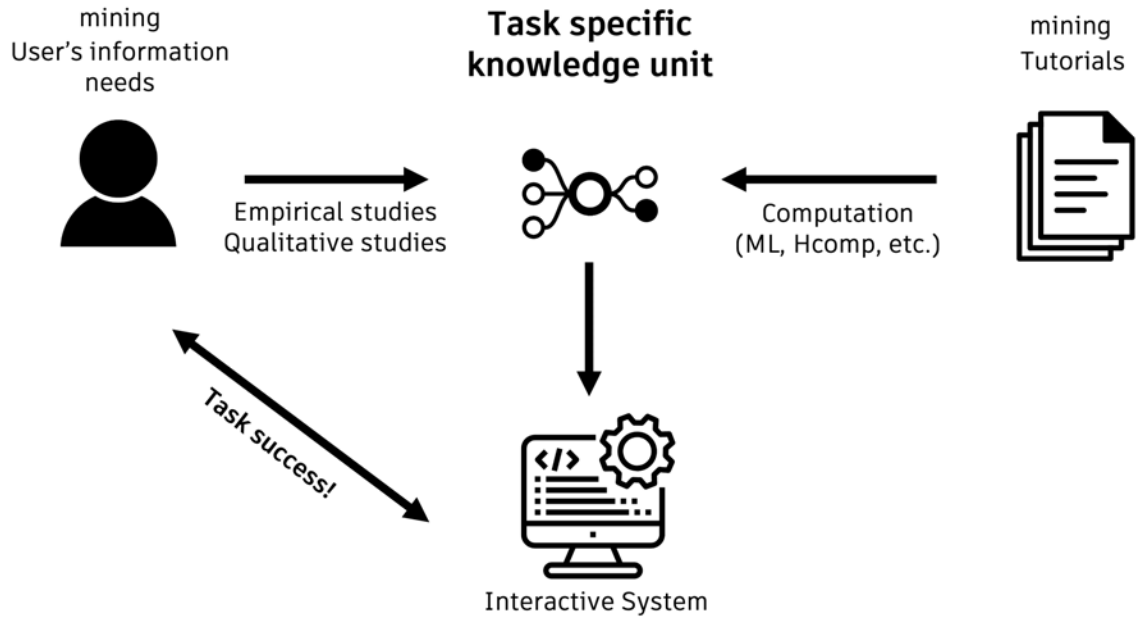


Figure 7.1: Task specific knowledge unit as the building block for novel interaction systems

Bridging the gap between user's mental model and the system design is at the very heart of Human–Computer Interaction research. This thesis makes a contribution by distilling users' information needs (one possible measure of users' mental model) and the computational building blocks (a component of system design), and thereby solving the mismatch in three commonly observed tutorial interactions, namely finding, navigating, and applying tutorials.

Techniques introduced in this thesis builds task specific knowledge units, which are composite structures of elements that make up a tutorial but augmented with user's task semantics. These task specific knowledge units serve as the glue between the computational unit in the system design which are often not task specific, and the users informational needs which are often too high level.

Interactive systems introduced in this thesis demonstrated that our novel "knowledge units" of complex semi-structured documents can help humans answer questions beyond information retrieval, i.e. find something in the pile. The methods introduced in the thesis brings what was previously unsearchable to being searchable, and provides interactions at the desired granularity that were previously unsupported. The conceptual key to this approach is defining task-relevant abstractions through data-driven (1) user-centered design and (2) inference algorithms. While developing this unique approach, I learned the following lessons about each.

7.1 User-centered Design

7.1.1 Access to Domain Expertise

To understand the specifics and nature of the task, having an access to domain expertise allows us to find deep insights about the user task that computational methods often cannot capture. For example, while building RecipeScape, access to cooking professionals allowed us to define "cooking structure".

Also, sometimes the tutorial data itself contains expert traces. For example, if the 3d modeling workflows are from either instructors or leading 3D designers, then there will be strategies and choices that are expert behaviors. But, it requires domain expertise to be able to spot them. That is why bottom-up data-driven approach usually requires human intervention.

Therefore, it is crucial to have access to domain experts.

7.2 Inference Algorithms

When aggregating a collection of sequential knowledge for inference, it is important to consider how tightly the results in each of the collection item are bounded.

If all tutorials in the collection lead to a strictly same goal, the aggregation becomes easier, because the tutorials or demonstrations will all have the same final state. In this case, we can leverage more complex structures like graphs that will allow us to perform computationally robust inferences (ex. graph algorithms) like in Workflow Graphs.

If the tutorials in the collection lead to a loosely similar goal, then clustering methods become more useful. Mining and understanding insights usually requires additional visualization techniques like in RecipeScape.

Chapter 8. Conclusion

8.1 Summary of Contribution

This thesis contributes three novel interactive systems, ReceipScape, RubySlippers, and Workflow Graphs in which the task specific knowledge units powering the systems are tree structure capturing procedural semantics in cooking, information balanced video segments with referenceable objects and actions, and unified subgoal structures in 3D modeling workflows, respectively. The three systems introduced provide effective alternative interaction techniques for finding, navigating, and applying tutorials.

8.1.1 Interactive Systems

Recipescape, allowed users to understand the "landscape" of all recipes related to a search query like "chocolate chip cookie". It also allowed users to visually search recipes which users found more convenient to use than typical list of search results interface. It allowed users to form hypothesis about "groups" of recipes while also Allowing users to validate their hypothesis.

RubySlippers allowed users to naviate using content-based referencing in voice user interface. The number of queries used were significantly less than that of temporal referencing, so even with parsing delays users didn't feel stressed or tedious. It also allowed users to effectively navigate when temporal referencing is difficult Users did not have much domain knowledge – no prior knowledge about the structure of the task, like navigating long videos, unwatched scenes, and multiple videos.

Workflow graphs contributed a conceptual approach for representing the different means by which a fixed goal can be achieved in feature rich software, based on recordings of user demonstrations, and has demonstrated a scalable pipeline for constructing such a represen- tation for 3D modeling software. It has also presented a range of applications that could leverage this representation to support users in improving their skill sets over time. Overall, we see this work as a first step toward enabling a new generation of help and learning systems for feature-rich software, powered by data-driven models of tasks and workflows.

8.2 Future Work

8.2.1 Workflow Graph as Knowledge Units

Our focus in Workflow Graph research is on enabling novice-to-expert transitions. This is one of the largest problems in software learning research, which has been widely studied but never fully addressed for high level workflows. In brief, learning design software is not just about learning how to do something, it's also about learning how to do something *better*. W-graphs has the potential to be a significant step toward addressing the novice-to-expert transition problem for high-level workflows, because it can capture diverse workflows including novice and expert methods, and methods applicable in different situations. An initial learner of a design software application may not benefit from many different workflows.

Also, undos and redos can be redesigned to support on sub-workflow level, not at the individual command level. This allows the system to recommends backtracing as a feasible next step to route the user to a more efficient workflow path.

Ultimately, one could imagine a single knowledge graph being created for each application, representing all the different tasks and workflows used in that software, and the relationships between them. This could then feed general help and learning techniques that provide feedback for any workflow a user is performing, provided that it has some relationship to those represented in the graph.

Also, W-graphs could also provide insights to the designers of software applications, which could be used to refine their interfaces and improve learnability. For example, if a W-graph produced by many novice users performing a task exhibited a high degree of branching, that might indicate that the software interface was not adequately supporting the workflow for performing that task. Likewise, producing a W-graph based on a sample of the user base could capture current practices and whether certain new workflows are being adopted or overlooked by the user community.

8.2.2 Causal Knowledge Units - Models in HCI

Models used to solve problems in HCI can be either empirical or computational. Empirical models try to explain the variability in the observed data, i.e. traces of human interaction behavior, whereas computational models try to explain the underlying mechanism of how the data is generated (mechanistic or computational explanation of the interaction decisions). In HCI research, “computation” plays a vital role as a means for obtaining understandings through empirical models. Such understandings have served as references and guidelines for designers to build more “human-centered” interfaces.

For example, this thesis primarily focused on the usages of empirical models that represent the compositionality of user tasks as observed in naturally crowdsourced tutorials and demonstrations. I developed techniques for extracting the nonlinear compositions of semantic substrates such as task-specific domain knowledge, constraints, and users’ strategies in large scale tutorials and demonstrations.

However users often do not know what they want. In other words, user goals are uncertain and dynamic. Because it is extremely difficult to empirically understand “causal relationships” from observational data, using data-driven insights to drive design heuristics - called data-driven design - is likely to be inadequate when incorporating the unknown uncertain dynamic user goal into the interface design problem.

Computational models have advantages over empirical counterparts. Computational models algorithmically describe relationships between the different quantities under observation. They describe the underlying process of human planning and interaction decisions. They inherently possess explainability and prediction of decisions. By design, they are able to “explain themselves”, which brings us one step closer to interpretable interfaces. And from the perspective of computational models, I’d like to argue that a human-computer interaction is a sequential decision-making problem for the user, and also for the system.

There are three promising methods. *METHOD 1.* “A user” can be included as a constraint in the objective function which a machine learning algorithm will try to optimize for. This will allow the user to be part of the probabilistic model of the world a machine learning algorithm will try to recover the underlying distribution of. While the expressive power of a “user” is quite limited in this method, there is a rich body of previous work in areas such as “Safe RL” and “Robust MDP”.

METHOD 2. "Cognitive Crash Dummies" and “Active inference” - Combining computational cognitive

models with Bayesian optimization to learn and optimize the parameters of the model enables us to build user simulators. These simulators can replace expensive user studies. More importantly, this facilitates researchers to conduct scientific experiments that are replicable and generalizable.

METHOD 3. An "interaction sequence" can also be modeled as a multi-agent shared goal optimization problem. Both the user and the interface are cooperative agents, but they are working towards a share goal. However, it is important to factor in the fact that they possess different computational bounds and capabilities. This is more directly involving users in the inference.

I wish to explore the design of interfaces for browsing and searching as the medium for collecting user behavioral data and for evaluating these methods. Such an interface should assist users in learning about their goals and in understanding the underlying mechanisms of the interface by allowing users to plan, form hypotheses and experiment. The interface should be able to make active inferences about the user's pursuit.

Making "AI" "human-centered" is a multifaceted problem. But in the context of building AI systems that can explain themselves, interactivity is vital. For interactivity to "make sense" for the users, I argue that we need to computationally understand usability. In order to do so, advances in both sequential decision-making models, and as well as in cognition-driven computational interface design are vital hand-in-hand.

Bibliography

[1] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. 2011. Flavor network and the principles of food pairing. *Scientific reports* 1 (2011). DOI:http://dx.doi.org/10.1038/srep00196

[2] David Akers, Robin Jeffries, Matthew Simpson, and Terry Winograd. 2012. Backtracking Events As Indicators of Usability Problems in Creation-Oriented Applications. *ACM Trans. Comput.-Hum. Interact.* 19, 2 (July 2012), 16:1–16:40. DOI:http://dx.doi.org/10.1145/2240156.2240164

[3] Abir Al-Hajri, Gregor Miller, Matthew Fong, and Sidney S Fels. 2014. Visualization of personal history for video navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1187–1196.

[4] Paul André, Aniket Kittur, and Steven P Dow. 2014. Crowd synthesis: Extracting categories and clusters from complex data. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 989–998.

[5] Zahra Ashktorab, Mohit Jain, Q Vera Liao, and Justin D Weisz. 2019. Resilient chatbots: repair strategy preferences for conversational breakdowns. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[6] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. 2002. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings. IEEE International Conference on Multimedia and Expo*, Vol. 1. IEEE, 705–708.

[7] Ira D Baxter, Aaron Quigley, Lorraine Bier, Marcelo Sant'Anna, Leonardo Moura, and Andrew Yahin. 1999. CloneDR: clone detection and removal. In *Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering*. 111–117.

[8] Morteza Behrooz, Sarah Mennicken, Jennifer Thom, Rohit Kumar, and Henriette Cramer. 2019. Augmenting Music Listening Experiences on Voice Assistants.. In *ISMIR*. 303–310.

[9] Derya Birant and Alp Kut. 2007. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering* 60, 1 (2007), 208–221.

[10] John Bluis and Dong-Guk Shin. 2003. Nodal distance algorithm: Calculating a phylogenetic tree comparison metric. In *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*. IEEE, 87–94. DOI:http://dx.doi.org/10.1109/bibe.2003.1188933

[11] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[12] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *ACM sigmod record*, Vol. 29. ACM, 93–104.

[13] John M. Carroll. 1990. *The Nurnberg funnel: designing minimalist instruction for practical computer skill*. MIT Press.

[14] John M. Carroll and Mary Beth Rosson. 1987. Paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. MIT Press, 80–111. http://dl.acm.org/citation.cfm?id=28446.28451

[15] Richard Catrambone. 1998. The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General* 127, 4 (1998), 355.

[16] Minsuk Chang, Léonore V Guillain, Hyeungshik Jung, Vivian M Hare, Juho Kim, and Maneesh Agrawala. 2018. RecipeScape: An Interactive Tool for Analyzing Cooking Instructions at Scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 451.

[17] Minsuk Chang, Vivian M Hare, Juho Kim, and Maneesh Agrawala. 2017. Recipescape: Mining and analyzing diverse processes in cooking recipes. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1524–1531. DOI:http://dx.doi.org/10.1145/3027063.3053118

[18] Minsuk Chang, Anh Truong, Oliver Wang, Maneesh Agrawala, and Juho Kim. 2019. How to design voice based navigation for how-to videos. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–11.

[19] Ming-Ming Cheng, Shuai Zheng, Wen-Yan Lin, Vibhav Vineet, Paul Sturgess, Nigel Crook, Niloy J Mitra, and Philip Torr. 2014. ImageSpirit: Verbal guided image parsing. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–11.

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

[21] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012b. MixT: Automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*. ACM, 93–102.

[22] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1999–2008.

[23] Eric Corbett and Astrid Weber. 2016. What can I say? Addressing user experience challenges of a mobile voice user interface for accessibility. In *Proceedings of the 18th international conference on human-computer interaction with mobile devices and services*. 72–82.

[24] GM Crippen. 1978. Note rapid calculation of coordinates from distance matrices. *J. Comput. Phys.* 26, 3 (1978), 449–452. DOI:http://dx.doi.org/10.1016/0021-9991(78)90081-5

[25] Chris Crockford and Harry Agius. 2006. An empirical investigation into user navigation of digital video using the VCR-like control set. *International Journal of Human-Computer Studies* 64, 4 (2006), 340–355.

[26] Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. 2011. MeshFlow: Interactive visualization of mesh construction sequences. *ACM Trans. Graph.* 30, 4 (July 2011), 66:1–66:8. DOI:http://dx.doi.org/10.1145/2010324.1964961

[27] Wei Ding and Gary Marchionini. 1998. *A study on video browsing strategies*. Technical Report.

[28] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. 2008. Video browsing by direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 237–246.

[29] Gregory Druck. 2013. Recipe attribute prediction using review text as supervision. In *Cooking with Computers 2013, IJCAI workshop*.

[30] Steven M Drucker, Georg Petschnigg, and Maneesh Agrawala. 2006. Comparing and managing multiple versions of slide presentations. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM, 47–56. DOI:http://dx.doi.org/10.1145/1166253.1166263

[31] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. 2017. Finding similar people to guide life choices: Challenge, design, and evaluation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 5498–5544. DOI:http://dx.doi.org/10.1145/3025453.3025777

[32] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[33] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 605–613.

[34] Adam Fourney, Ben Lafreniere, Parmit K. Chilana, and Michael Terry. 2014. InterTwine: Creating interapplication information scent to support coordinated use of software. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, 10 pages.

[35] Adam Fourney, Richard Mann, and Michael Terry. 2011a. Characterizing the usability of interactive applications through query log analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1817–1826.

[36] Adam Fourney, Richard Mann, and Michael Terry. 2011b. Query-feature Graphs: Bridging User Vocabulary and System Functionality. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 207–216. DOI:http://dx.doi.org/10.1145/2047196.2047224

[37] C Ailie Fraser, Tricia J Ngoon, Mira Dontcheva, and Scott Klemmer. 2019. RePlay: Contextually Presenting Learning Videos Across Software Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

[38] Elena L. Glassman, Aaron Lin, Carrie J. Cai, and Robert C. Miller. 2016. Learnersourcing Personalized Hints. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social*

Computing (CSCW '16). ACM, New York, NY, USA, 1626–1636. DOI:http://dx.doi.org/10.1145/2818048.2820011

[39] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph.* 28, 3 (July 2009), 66:1–66:9. DOI: http://dx.doi.org/10.1145/1531326.1531372

[40] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for Accelerating On-line Learning of Hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1591–1600. DOI:http://dx.doi.org/10.1145/1240624.1240865

[41] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A Survey of Software Learnability: Metrics, Methodologies and Guidelines. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 649–658. DOI:http://dx.doi.org/10.1145/1518701.1518803

[42] 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 (UIST '10)*. ACM, New York, New York, USA, 143–152. DOI: http://dx.doi.org/10.1145/1866029.1866054 ACM ID: 1866054.

[43] Philip J. Guo. 2015. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 599–608. DOI:http://dx.doi.org/10.1145/2807442.2807469

[44] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. 2012. DuploTrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 389–402.

[45] Dan Jackson, James Nicholson, Gerrit Stoeckigt, Rebecca Wrobel, Anja Thieme, and Patrick Olivier. 2013. Panopticon: A parallel video overview system. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 123–130.

[46] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. 2007. Deckard: Scalable and accurate tree-based detection of code clones. In *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 96–105. DOI:http://dx.doi.org/10.1109/icse.2007.30

[47] Hyoungwook Jin, Minsuk Chang, and Juho Kim. 2019. SolveDeep: A System for Supporting Subgoal Learning in Online Math Problem Solving. In *Proceedings of the SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM.

[48] Stephen C Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254. DOI: http://dx.doi.org/10.1007/bf02289588

[49] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. 2002. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering* 28, 7 (2002), 654–670. DOI:http://dx.doi.org/10.1109/tse.2002.1019480

[50] Masahiro Kazama, Minami Sugimoto, Chizuru Hosokawa, Keisuke Matsushima, Lav R Varshney, and Yoshiki Ishikawa. 2017. Sukiyaki in French style: A novel system for transformation of dietary patterns. *arXiv preprint arXiv:1705.03487* (2017).

[51] Juho Kim. 2013. Toolscape: enhancing the learning experience of how-to videos. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2707–2712.

[52] Juho Kim. 2015. *Learnersourcing: Improving Learning with Collective Learner Activity*. PhD thesis. Massachusetts Institute of Technology.

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

[54] Juho Kim, Philip J Guo, Daniel T Seaton, Piotr Mitros, Krzysztof Z Gajos, and Robert C Miller. 2014b. Understanding in-video dropouts and interaction peaks in online lecture videos. In *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 31–40.

[55] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J Guo, Robert C Miller, and Krzysztof Z Gajos. 2014c. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 4017–4026.

[56] Yea-Seul Kim, Mira Dontcheva, Eytan Adar, and Jessica Hullman. 2019. Vocal shortcuts for creative experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–14.

[57] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012a. Delta: a tool for representing and comparing workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1027–1036. DOI:http://dx.doi.org/10.1145/2207676.2208549

[58] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. 2012b. Delta: a tool for representing and comparing workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1027–1036. DOI:http://dx.doi.org/10.1145/2208516.2208549

[59] Nicholas A Kraft, Brandon W Bonds, and Randy K Smith. 2008. Cross-language Clone Detection.. In *SEKE*. 54–59.

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

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

[62] Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. Pixeltone: A multimodal interface for image editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2185–2194.

[63] Walter S Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P Bigham, and Michael S Bernstein. 2015. Apparition: Crowdsourced user interfaces that come to life as you sketch them. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1925–1934.

[64] Francis C Li, Anoop Gupta, Elizabeth Sanocki, Li-wei He, and Yong Rui. 2000. Browsing digital video. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 169–176.

[65] Wei Li, Tovi Grossman, and George Fitzmaurice. 2014. CADament: A Gamified Multiplayer Software Tutorial System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3369–3378. DOI:http://dx.doi.org/10.1145/2556288.2556954 event-place: Toronto, Ontario, Canada.

[66] Wei Li, Yuanlin Zhang, and George Fitzmaurice. 2013. TutorialPlan: automated tutorial generation from CAD drawings. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[67] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. 2013. Skillometers: Reflective Widgets That Motivate and Help Users to Improve Performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 321–330. DOI:http://dx.doi.org/10.1145/2501988.2501996

[68] Sana Malik, Fan Du, Megan Monroe, Eberechukwu Onukwugha, Catherine Plaisant, and Ben Shneiderman. 2015. Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, 38–49. https://doi.org/10.1145/2678025.2701407

[69] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.. In *ACL (System Demonstrations)*. 55–60. DOI:http://dx.doi.org/10.3115/v1/p14-5010

[70] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011. Ambient help. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2751–2760. DOI:http://dx.doi.org/10.1145/1978942.1979349

[71] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2012. Swift: reducing the effects of latency in online video scrubbing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 637–646.

[72] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2013. Swifter: improved online video scrubbing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1159–1168.

[73] Sarah McRoberts, Joshua Wissbroecker, Ruotong Wang, and F Maxwell Harper. 2019. Exploring Interactions with Voice-Controlled TV. *arXiv preprint arXiv:1905.05851* (2019).

[74] David Mogensen. 2015. I want-to-do moments: From home to beauty. *Think with Google* (2015).

[75] Christine Murad, Cosmin Munteanu, Leigh Clark, and Benjamin R Cowan. 2018. Design guidelines for hands-free speech interaction. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 269–276.

[76] Chelsea Myers, Anushay Furqan, Jessica Nebolsky, Karina Caro, and Jichen Zhu. 2018. Patterns for How Users Overcome Obstacles in Voice User Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 6.

[77] Chelsea M Myers. 2019. Adaptive suggestions to increase learnability for voice user interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*. 159–160.

[78] Cuong Nguyen and Feng Liu. 2015. Making software tutorial video responsive. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1565–1568.

[79] Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 1–12. DOI:http://dx.doi.org/10.1145/2047196.2047198

[80] Amy Pavel, Floraine Berthouzoz, Björn Hartmann, and Maneesh Agrawala. 2013. Browsing and analyzing the command-level structure of large collections of image manipulation tutorials. *Citeseer, Tech. Rep.* (2013).

[81] JB Phipps. 1971. Dendrogram topology. *Systematic zoology* 20, 3 (1971), 306–308. DOI:http://dx.doi.org/10.2307/2412343

[82] Florian Pinel and Lav R Varshney. 2014. Computational creativity for culinary recipes. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. ACM, 439–442. DOI:http://dx.doi.org/10.1145/2559206.2574794

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

[84] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. 2011b. Pause-and-play: Automatically Linking Screencast Video Tutorials with Applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 135–144. DOI:http://dx.doi.org/10.1145/2047196.2047213

[85] Martin Porcheron, Joel E Fischer, Stuart Reeves, and Sarah Sharples. 2018. Voice Interfaces in Everyday Life. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 640.

[86] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv preprint arXiv:1612.00593* (2016).

[87] Markus Rokicki, Eelco Herder, Tomasz Kuśmierczyk, and Christoph Trattner. 2016. Plate and prejudice: Gender differences in online cooking. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM, 207–215. DOI:http://dx.doi.org/10.1145/2930238.2930248

[88] Xin Rong, Adam Fourney, Robin N Brewer, Meredith Ringel Morris, and Paul N Bennett. 2017. Managing uncertainty in time expressions for virtual assistants. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 568–579.

[89] Chanchal K Roy, James R Cordy, and Rainer Koschke. 2009. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of computer programming* 74, 7 (2009), 470–495. DOI:http://dx.doi.org/10.1016/j.scico.2009.02.007

[90] Jeffrey Rzeszotarski and Aniket Kittur. 2012. CrowdScape: interactively visualizing user behavior and output. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 55–62. DOI:http://dx.doi.org/10.1145/2380116.2380125

[91] Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. 1978. A simplest systematics for the organization of turn taking for conversation. In *Studies in the organization of conversational interaction*. Elsevier, 7–55.

[92] Naruya Saitou and Masatoshi Nei. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution* 4, 4 (1987), 406–425. DOI:http://dx.doi.org/10.1093/oxfordjournals.molbev.a040454

[93] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning Cross-modal Embeddings for Cooking Recipes and Food Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. DOI:http://dx.doi.org/10.1109/cvpr.2017.327

[94] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery* 2, 2 (1998), 169–194.

[95] Christian Santoni, Claudio Calabrese, Francesco Di Renzo, and Fabio Pellacini. 2016. SculptStat: Statistical Analysis of Digital Sculpting Workflows. *arXiv preprint arXiv:1601.07765* (2016).

[96] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. 2011. Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2741–2750. DOI:http://dx.doi.org/10.1145/1978942.1979348

[97] Ben Shneiderman, Catherine Plaisant, and Bradford W Hesse. 2013. Improving healthcare with interactive visualization. *Computer* 46, 5 (2013), 58–66. DOI:http://dx.doi.org/10.1109/mc.2013.38

[98] Tiago Simas, Michal Ficek, Albert Diaz-Guilera, Pere Obrador, and Pablo R Rodriguez. 2017. Food-bridging: a new network construction to unveil the principles of cooking. *arXiv preprint arXiv:1704.03330* (2017).

[99] Tanmay Sinha, Patrick Jermann, Nan Li, and Pierre Dillenbourg. 2014. Your click decides your fate: Inferring Information Processing and Attrition Behavior from MOOC Video Clickstream Interactions. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. 3–14.

[100] Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering natural language commands in multimodal interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 661–672.

[101] Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. 2015. Building hierarchies of concepts via crowdsourcing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[102] Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *Journal of the ACM (JACM)* 26, 3 (1979), 422–433. DOI:http://dx.doi.org/10.1145/322139.322143

[103] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. 2012. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*. ACM, 298–307. DOI: http://dx.doi.org/10.1145/2380718.2380757

[104] Robert L Thorndike. 1953. Who belongs in the family? *Psychometrika* 18, 4 (1953), 267–276. DOI: http://dx.doi.org/10.1007/bf02289263

[105] Warren S Torgerson. 1952. Multidimensional scaling: I. Theory and method. *Psychometrika* 17, 4 (1952), 401–419. DOI:http://dx.doi.org/10.1007/bf02288916

[106] Kush R Varshney, Lav R Varshney, Jun Wang, and Daniel Myers. 2013. Flavor pairing in Medieval European cuisine: A study in cooking with dirty data. *arXiv preprint arXiv:1307.7982* (2013).

[107] Alexandra Vtyurina and Adam Fourney. 2018. Exploring the Role of Conversational Cues in Guided Task Support with Virtual Assistants. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 208.

[108] w3c. 2018. Web Accessibility Initiative. https://www.w3.org/ WAI/. (2018).

[109] Jayne Wallace, John McCarthy, Peter C Wright, and Patrick Olivier. 2013. Making design probes work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3441–3450.

[110] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018a. 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*. ACM, 285.

[111] Xu Wang, Benjamin J. Lafreniere, and Tovi Grossman. 2018b. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *CHI*.

[112] Yiting Wang, Walker M White, and Erik Andersen. 2017. PathViewer: Visualizing Pathways through Student Data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 960–964. DOI:http://dx.doi.org/10.1145/3025453.3025819

[113] Jacob Whitehill and Margo Seltzer. 2017. A Crowdsourcing Approach to Collecting Tutorial Videos–Toward Personalized Learning-at-Scale. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*. ACM, 157–160.

[114] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z. Gajos, Walter S. Lasecki, and Neil Heffernan. 2016. AXIS: Generating Explanations at Scale with Learnersourcing and Machine Learning. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale (L@S '16)*. ACM, New York, NY, USA, 379–388. DOI:http://dx.doi.org/10.1145/2876034.2876042

[115] Lisa Yan, Nick McKeown, and Chris Piech. 2019. The PyramidSnapshot Challenge: Understanding student process from visual output of programs. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19). ACM, New York, NY, USA. https://doi. org/10.1145/3287324.3287386*.

[116] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3861–3870.

[117] Longqi Yang, Yin Cui, Fan Zhang, John P Pollak, Serge Belongie, and Deborah Estrin. 2015. Plate-click: Bootstrapping food preferences through an adaptive visual interface. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 183–192. DOI: http://dx.doi.org/10.1145/2806416.2806544

[118] Ziheng Yang. 1997. PAML: a program package for phylogenetic analysis by maximum likelihood. *Bioinformatics* 13, 5 (1997), 555–556. DOI:http://dx.doi.org/10.1093/bioinformatics/13.5.555

[119] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A Scalable Active Framework for Region Annotation in 3D Shape Collections. *SIGGRAPH Asia* (2016).

[120] Ning Yu, Desislava Zhekova, Can Liu, and Sandra Kübler. 2013. Do good recipes need butter? Predicting user ratings of online recipes. In *Proceedings of the IJCAI Workshop on Cooking with Computers, Beijing, China*.

[121] Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing* 18, 6 (1989), 1245–1262. DOI:http://dx.doi.org/10.1137/0218082

[122] Kaizhong Zhang, Rick Statman, and Dennis Shasha. 1992. On the editing distance between unordered labeled trees. *Information processing letters* 42, 3 (1992), 133–139. DOI:http://dx.doi.org/10.1016/0020-0190(92)90136-j

Acknowledgment

Like the saying goes, it really takes a village to raise a child. My PhD journey and this thesis would not have been possible without support from family, advisors, mentors, colleagues and friends. Specifically, I would like to express eternal gratitude to:

- my wife Anne, who is my best friend and the biggest supporter;
- my daughter Juha, who is my eternal source of energy and happiness;
- my parents Seongdae, Jeonghee, my in-laws Sunggul and Namhee for their unconditional love;
- my advisor Juho Kim for not only helping me become a better researcher, but also a better person. ;
- the world's best dissertation committee - Meeyoung Cha, Mira Dontcheva, Elena Glassman, and Alice Oh;
- my internship mentors, who have taught me so much in and outside of research - Anh Truong, Oliver Wang, Ben Lafreniere, Tovi Grossman, Adam Fourney;
- Antti Oulasvirta for his mentorship, especially encouraging me to follow exciting ideas and to never settle;
- my coauthors and collaborators of this thesis;
- Current and past members of the KIXLAB.