Saga: Understanding Stories in Mobile App Reviews

HUI GUO*, Quora Inc., USA
MUNINDAR P. SINGH, North Carolina State University, USA

Online storytelling is an essential channel for users to express their experiences and opinions and therefore influence online society. Yet, despite its importance, approaches to story understanding on social media have not advanced sufficiently. Specifically, current approaches can carry out high-level, aggregate analyses on a corpus of stories but do not provide a way of understanding individual stories. We consider a major source of social behavior, app reviews, which surprisingly are rarely studied in social media research. We observe that app reviews often contain one or more stories. These stories exhibit complex structures and are often presented via events that are not placed in their natural order.

Accordingly, we introduce Saga, an approach that carries out a deep analysis of the event-based structures and substructures arising in app reviews. Saga's main contribution is how it goes beyond the state of the art in identifying fine-grained story (sub)structures. In addition, it supports querying stories (and their containing app reviews) according to these (sub)structures. These specific (sub)structures help identify stories that serve different information goals. Saga is evaluated both computationally on a publicly available data source and via a human study validating the helpfulness in addressing various information goals.

CCS Concepts: • **Applied computing** → *Document searching*; • **Software and its engineering** → *Maintaining software*; • **Information systems** → **Structure and multilingual text search**.

Additional Key Words and Phrases: Stories on Social Media, App Reviews, Events, Story Structures, Natural Language Processing

ACM Reference Format:
Hui Guo and Munindar P. Singh. 2025. Saga: Understanding Stories in Mobile App Reviews. *ACM Trans. Web* 0, 0, Article 0 (January 2025), 22 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Review writing is a widely occurring user behavior on online social networks and media platforms. Providing feedback via online reviews is an important channel through which users of mobile apps and products can affect the behaviors of app developers and product manufacturers, as well as of other users. In the area of software engineering, app reviews have been frequently analyzed for information that is helpful to app developers, such as bug reports and feature requests [26]. The computational social media community, e.g., Sun et al. [43], often focuses on the sentiments online reviews express toward their target or self-descriptions of the reviewers. We have found that online reviews contain a vast variety of stories about reviewers' interactions with the apps or products. These stories are rich in information and should be understood coherently instead of being stripped into pieces of text based on their usefulness.

*Work performed while at NC State University.

Authors' addresses: Hui Guo, Quora Inc., Mountain View, California, USA, hguo5@alumni.ncsu.edu; Munindar P. Singh, mpsingh@ncsu.edu, North Carolina State University, Raleigh, North Carolina, USA.

ACM 1559-1131/2025/1-ART0
https://doi.org/XXXXXXX.XXXXXXX

From the perspective of software engineering, it may be pragmatic to focus on the translation from user-generated text snippets into formatted requirements. Extensive research has been conducted on data mining techniques for the extraction of useful information that supports software engineering activities from app reviews, including user intentions, discussion topics, requests for new features, and reports of bugs [8, 11, 26, 42, 47]. Such information can be directly mapped to functional or nonfunctional requirements and usable by software developers. Although some review writers are succinct when offering their opinions, reports, or requests, many others describe more intricate interaction stories. In addition to software requirements, these stories provide affluent information regarding user behaviors [16] and rationales [23], which are essential for user-centric software development. Mining such information requires an understanding of the narratives and the organization of different parts in user-written stories. In this study, we have found that stories with specific structures are helpful toward different developers' goals, such as understanding app problems, user retention, and user expectations.

Researchers in social media recognize the importance of understanding stories or narratives. However, one challenge in the fine-grained understanding of user-generated stories on online social platforms at large is their complexity. Authors of such stories hold different underlying psychosocial motivations and do not conform to fixed narratives. Broadly, a story is a description of what transpired in some purportedly objective reality and a narrative is how a story is presented. Previous work on narrative understanding in social media focuses on a general study of narratives in aggregate. For example, Habib and Nithyanand [17] extract high-level themes from stories about COVID. Xi and Singh [45] analyze narratives for how they position their protagonists (authors) and antagonists. To that end, they employ several features for analysis, including their grammatical structure and various semantic elements. In the same spirit, Giorgi et al. [14] capture event chains based on groupings of verbs that arise in different positions in the stories across the corpus.

Our work emphasizes the structures of the discovered stories in the domain of app reviews. The structure of a story captures the order in which events occur, which may differ from the narrative (i.e., discourse) order. Analyzing story structures in online stories gives us a clearer understanding of what types of stories transpired in user-product interactions. We target mobile app reviews in this work, which are a rich source of interaction stories between a reviewer (as a user) and an app. First, app reviews offer a large number of stories in a tractable setting for research, i.e., interaction stories with an app. We can largely avoid the complexity of human-human interactions and the associated psychosocial factors. Second, app reviews exhibit structurally diverse narratives, as each reviewer is free to express their own experiences and opinions. Finally, understanding such stories may help app developers improve their products.

We construe a *story* as a sequence of events in order of occurrence; here, each event may be of a different type. Example 1 shows a review snippet for the Snapchat app[1] from Apple App Store, which describes a story, consisting of several events, each marked with an underline with its type.

Example 1

★☆☆☆☆ username1, 06/25/2014

Wifi?

I'm trying to sign up$_{\text{INTENTION}}$ and on the part where you write your username, I press done after I type it$_{\text{ACTION}}$ and it brings up a message saying to check my connection$_{\text{BEHAVIOR}}$. ... I've checked my connection$_{\text{REACTION}}$ and I've re-downloaded the app$_{\text{REACTION}}$. It won't work$_{\text{BEHAVIOR}}$!! Please fix it.

[1] https://apps.apple.com/us/app/snapchat/id447188370

We define a story *structure* as a sequential pattern of event *types*. The order in which the event types occur is a strong indicator of the story's narrative structure. We target five event types that are relevant for app reviews, defined in Section 3.2. Under this framework, each story in an app review can be represented by its structure. Example 1 exhibits a structure of ⟨Intention → Action → Behavior → Reaction → Behavior⟩ (IABRB). Additionally, we define a *substructure* as a sequential pattern that is part of a story structure, e.g., ⟨Intention → Action⟩ or ⟨Behavior → Reaction⟩. A substructure can represent a sequence of events that constitute a meaningful snippet of a story.

Therefore, we can focus research on coherent user stories, represented by their structures, rather than discrete story elements. Stories with different structures may serve different goals for the storytellers and can be useful to different app review analysts. Structures like ⟨Action → Behavior⟩ are indicative of failures [16]. Structures like ⟨Intention → Action⟩ may provide insights into the reviewer's mental model and expectations when using the app, as they describe situations where user actions are triggered by their intentions. Structures ⟨Behavior → Reaction⟩ may help indicate technology acceptance or rejection [12]. In fact, the structure of ⟨Intention → Action → Behavior → Reaction⟩ is the basic form of user-app interactions and is instrumental in studying the confluence of human and machine agency [34, 39]. Stories with certain structures can be further analyzed for directions of app improvement or refined into software requirements.

We propose Saga, a framework for analyzing story structures that arise in app reviews and retrieving stories of the specified structures. Saga includes three main steps. First, it leverages NLP techniques to extract events from app reviews and classify their types. Second, Saga adopts heuristics and trains classification models to learn the relations between events, and automatically identify the sequential order of events in a story. Third, based on the extracted stories and their structures, it enables the search of app reviews and user stories by their story structures. We conduct frequent pattern mining to collect the most frequent story structures and substructures in app reviews.

Grasping the structures of stories in app reviews is nontrivial. It is not uncommon for app reviews to mix event descriptions with additional text that is not part of the stories, such as expressions of emotion and call-outs to the developers. Moreover, some events may not contribute to the understanding of user-app interaction, such as hypothetical events. Once we extract the events of interest, combining them into stories may not be straightforward, as they may be out of order or belong to different stories. For a systematic analysis of stories in app reviews, Saga addresses the following research questions.

$RQ_{extract}$ How effectively can we extract events and determine their types in app reviews?
RQ_{relate} How effectively can we identify relations between events to sequence them into stories?
$RQ_{collect}$ What kind of story structures and substructures are the most common in app reviews?

We address **$RQ_{extract}$** and **RQ_{relate}** by reporting Saga's accuracy in classifying event types and identifying event relations. We address **$RQ_{collect}$** by presenting the common story structures and substructures that Saga mined from the user stories in app reviews. By discovering common story structures and substructures in stories, Saga sheds new light on a little-studied type of social media behavior. Additionally, we show the effectiveness of Saga from the perspective of software development. We conduct a small-scale human study asking annotators to rate the helpfulness of the stories of specific structures for understanding app problems, user retention, and user expectation.

Saga contributes to the literature of software engineering by enabling requirement elicitation and user understanding from coherent user stories instead of fragmented text pieces. By identifying

fine-grained story structures, Saga takes the first step toward the understanding of user behaviors on online platforms through stories.

Organization. The rest of the paper is organized as follows. Section 2 describes the research work on app review and event relations. Section 3 introduces the targeted data and our method in Saga. Section 4 demonstrates the results of our method. Section 5 discusses the merits of Saga, as well as limitations and potential future work to address them. Section 6 concludes this paper.

2 RELATED WORK

We now introduce related work on events, app reviews, and user-app interactions.

2.1 Event Relations

We target user stories described in app reviews. Unlike traditional stories, app reviews include a large amount of text that is not part of a story, and may not describe events sequentially. To combine related events into stories, we need to determine the relations between events, such as their sequencing. We now introduce studies on event relations.

Earlier studies target temporal relations of events based on how they appear in text, using heuristics or unsupervised methods. Mani et al. [27] use rules and axioms to infer temporal relations between events. Mirroshandel and Ghassem-Sani [31] extract temporally related events from news articles based on event features such as tense, polarity, and modality, as well as event-event features, such as prepositional phrases. Ning et al. [33] build a probabilistic knowledge base by extracting temporal relations from news articles. They argue that other types of extraction of temporal relations can benefit from such prior knowledge about the temporal order events usually follow.

Causal relations between events can be inferred based on events' temporal relations. Hu and Walker [20] extract frequent event pairs from movie scripts and infer their *causal potentials* [1] by comparing the frequencies of a pair and its reverse pair. Based on similar ideas, Hu et al. [19] extract and infer fine-grained event pairs that are causally related from blogs and film descriptions. Ning et al. [32] propose a framework that jointly reasons about and extracts temporal and causal relations between annotated events from texts and improves the extraction of both relations.

Some studies seek to understand the relations between events at a deeper level and with reduced reliance on context. Rashkin et al. [36] investigate the intents and reactions of a single event. They provide Event2Mind, a dataset of 25,000 events along with commonsense intents and reactions, built via crowdsourcing. ATOMIC [38] extends Event2Mind by incorporating more *inferential dimensions* along which follow-up or preceding events can be inferred and includes commonsense knowledge for 24,000 base events. Both studies propose models to infer events based on their relations. BERT [7] is a popular transformer-based language model that provides a solution for next sentence prediction (NSP): to predict whether a second sentence is the next sentence of the first. Though BERT's NSP model cannot be directly used to predict relations between events, it shows that concatenation is a viable way of dealing with two input sentences. We borrow ideas from the above studies and adopt both heuristics and classification models to determine the relation between events.

2.2 App Review Analysis

Although app reviews may include valuable information for app developers, not all app reviews are informative [11, 25]. Earlier studies targeted the identification and classification of useful reviews, leveraging text classification techniques on entire reviews. Maalej and Nabil [26] introduce four types of app reviews, i.e., bug reports, feature requests, user experiences, and text ratings, and classify app reviews using NLP techniques. Ciurumelea et al. [4] define a taxonomy of specific

categories regarding mobile apps, such as performance, resources, battery, and memory. Based on this taxonomy, they leverage NLP techniques in an automatic method for organizing reviews. McIlroy et al. [29] identify the issues raised by a negative (one-star or two-star) review. Dąbrowski et al. [10] analyze users' requests and sentiments on different features of an app. Garg et al. [13] comb through app reviews with NLP techniques and search for reviews that expose app functionalities that enable misuse. Such findings can facilitate automated misuse audits for apps. They have identified numerous mobile apps on the Apple App Store that can be exploited by rogue users.

Identifying reviews that contain useful information saves time on manual filtering for developers, but taking advantage of the large number of whole reviews remains cumbersome. Recent work focuses on the extraction of useful information in a compact form. Di Sorbo et al. [9] introduce a summarizer for user reviews that condenses the information residing in the large number of reviews a popular app receives. Such a summarizer substantially reduces the analysis time for developers. Jha and Mahmoud [21] automatically capture nonfunctional requirements (NFRs) in categories such as performance and usability from app reviews. Truelove et al. [44] leverage topic modeling to identify user issues, such as connectivity, timing, and updates, in reviews for Internet of Things (IoT) based apps.

2.3 User-App Interaction

The interaction between humans and apps has gained increasing interest among researchers in both software engineering and social studies. Researchers have started to look at user-app interactions as a dynamic process, in which both humans and machinery influence each other and evolve together. App reviews are very well situated in this process, and provide a rich source for the studies of different aspects. Grace et al. [15] analyze the contextual details users provide in the reviews of a personal safety app and suggest rationals for app updates with new features. Liu et al. [24] introduce app reviews into goal-oriented requirements engineering and mine user sentiment from reviews. They provide insights into optimizing the evaluation strategy of apps by evaluating the level of goal achievement and the parts of the apps that need to be improved. They aim to help the apps retain old users and attract new ones. Guo and Singh [16] extract user-action app-problem event pairs from app reviews. They have found that one common theme in negative app reviews is user actions triggering app problems. Guo and Singh collect mini-stories that are easy to read and analyze for a developer who wishes to fix their app's problems. However, their work misses other types of stories and fails to provide insights into the understanding of users' review writing behaviors. Saga extends their work by considering more event types and story structures.

Research has been conducted on how apps influence human behaviors as well, as user-app interactions are often of negotiated narrative structures. Siles et al. [39] investigate the process of algorithm awareness, in which users become aware of an app's algorithms. They analyze the various reactions in response; users may try to train the app, manage their expectations, or show forms of rejection to the app. AI-powered apps may further show agency and interact with human agency. Kang and Lou [22] study users' reception to their personalized experience brought about by AI algorithms that exhibit machine agency. They have found that users may deliberately behave in a way that makes the algorithms cater more to their needs. Obreja [34] analyze how users make sense of their interaction with TikTok's algorithms. They posit that TikTok users legitimize the presence of institutional actors, which is a form of agency negotiation between users and machines. Saga does not focus on a specific aspect of user-app interaction but attempts to understand the common structures interaction stories as presented in app reviews.

3 METHOD

Saga includes three components, as shown in Figure 1. First, Saga extracts events from targeted app reviews and identifies their types. Second, Saga combines heuristics and a classifier to determine the relations between two events. Events are ordered and sequenced to form structured stories. Third, Saga enables the search on the structured stories and collects stories matching a query. The framework also mines frequent story structures and substructures.

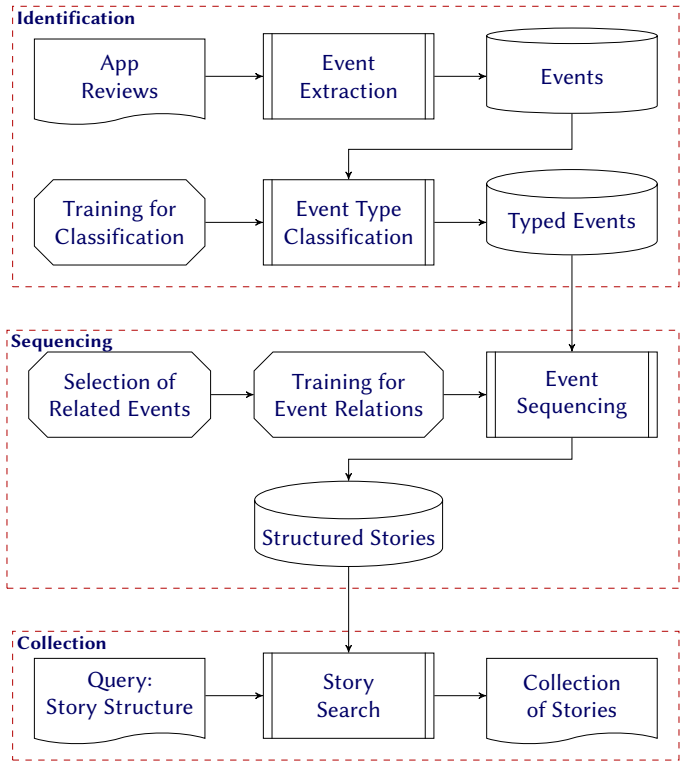

Fig. 1. An overview of Saga.

3.1 Dataset: Targeted App Reviews

We collected app reviews received by 182 apps from the period of 2008-07-10 to 2019-02-04, by crawling the app reviews pages on Apple App Store.[2] Based on the intuition that negative reviews describe what went wrong with a user interaction, we expect them to include richer stories [16, 29]. In total, we collected 2,118,942 negative app reviews. On the Apple App Store, user names of the reviews are publicly available. However, our research does not involve a study of the storytellers. All identifiers have been removed from the dataset and excluded in our subsequent operations.

3.2 Event Identification

To identify target events from app reviews, we extract event phrases from the reviews using NLP techniques and determine their types through classification. We conducted manual labeling to build a training set for the classifier.

[2] https://apps.apple.com/us/genre/ios/id36

3.2.1 Event Extraction. We define an *event* as something described by an *event phrase* [16, 36, 38], namely, a piece of text (typically a sentence or a clause) rooted in a target verb [6]. We adopt *Part-of-Speech (POS) tagging* [37] and *dependency parsing* [6] to extract event phrases.

A POS tagger identifies the verbs in a sentence, but not all verbs and their corresponding verb phrases constitute meaningful events. We determine target verbs, as well as the event phrases rooted in them, based on dependency parsing. We consider only verbs marked ROOT, advcl (adverbial clause modifier), or conj (conjunct) because these verbs are likely to be the root of an event phrase. Note that all words in a sentence have dependencies leading to a ROOT word, and a word may have dependencies leading to multiple verbs. Therefore, we do not consider a word if it is marked as part of another event phrase. We adopt the POS tagger and dependency parser implemented in spaCy[3] in this step.

3.2.2 Manual Labeling. To determine the types of the extracted events, we need to label a set of events to train a classifier. We consider the following five types of events as *target* events, and all other events as the **Nontarget** type (**N**):

- **(I) Intention:** The user intends to fulfill a need or bring about an app behavior. We assume that as of the reference time of the utterance [5], the user has not taken an action or achieved the intention.
- **(A) Action:** The user takes or tries taking an action within the app. As of the reference time of the utterance, the user has taken at least part of the described action.
- **(B) Behavior:** The app exhibits a behavior, typically in reaction to the user's action. This definition includes the absence of a behavior, such as "app does not respond to user's action."
- **(R) Reaction:** The user reacts to an app behavior, including attempts to solve a problem and being forced to take an action in response to a problem.
- **(C) Context:** Contextual events related to the above types, which are typically included to bolster the reviewer's claim.

The classification of certain events, such as Reaction, may rely on the context in which they appear. Thus, during the annotations, each event phrase is shown along with the text covering its preceding and following events.

User-generated stories are a hodgepodge of events, and determining event types can be challenging. In Table 1, we list guidelines for some commonly seen event themes that are difficult to classify. However, we leave the final verdict to the annotators.

Based on this classification, we conducted a small-scale investigation on a dataset of 300 randomly selected events. The number of events in each type is shown in Table 2. It is notable that the distribution of event types is skewed.

To sample a more balanced dataset for the final annotation, we leveraged the 300 labeled data points as a seed dataset. We first encoded each event into a vector using the Universal Sentence Encoder (USE) [3], a transformer-based sentence embedding that captures rich semantic information. We then determined the potential type of each event using k-nearest neighbors (KNN, we use $k = 7$), where the distance between two events is determined by the cosine similarity of their USE vectors. We randomly sampled 500 events in each type, including the **Nontarget** type, resulting in a dataset of 3,000 events.

We adopted crowdsourcing on Amazon Mechanical Turk[4] to obtain the labels for the types of the 3,000 events. All identifiers have been removed from the event phrases. Instructions for annotation included the definitions of the event types and respective examples. Each event was

[3] https://spacy.io/
[4] https://www.mturk.com/

Table 1. Annotation guidelines for ambiguous events.

Event Type	Event Theme
Action	*I accidentally did something in app*
	Other users did something in app that affected me
Behavior	*Nothing changed/helped after I tried something*
	I have to do something because of app design
Reaction	*I have to do something because an app problem*
	I am done with this app because of problem
	I will do something in response to problem
	I refuse to do what was asked
	I am still waiting for things to change
Context	*Other people are having the same issue*
	I get something (a message, a notification, etc.) from/in app (and it is not a problem)
	App is updated
	I forgot to do something
	I decided to use this app
	I used to do something about the app
	Events about device type and OS version
	Contextual events about security issues
Nontarget	*I get the developers are trying to do something*
	Opinion: *Developers need to do something*
	Opinion: *I used to love this app*
	Opinion: *I do not want something*
	Requests or questions
	Rating related events
	Hypothetical events

labeled by two crowd workers, and any disagreements were resolved by one of the authors. Each crowd worker was paid $0.02 for labeling one event, which took a median of 18 seconds. This study was approved by our university's Institutional Review Board (IRB). This dataset is available at https://github.com/hguo5/Saga for reproducibility. The final distribution of this labeled dataset, as shown in Table 2, is more balanced than the seed dataset.

3.2.3 Event Type Classification. Determining the type of an event is a six-class classification. We first convert event phrases into vectors using the USE, and then apply classification models on the vectors. The classification models that we experiment with include k-nearest neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT), and Multi-Layer Perceptron (MLP).

Additionally, as the context of an event phrase is considered during annotation, we experiment with ways to leverage this contextual information. We consider the two segments of text that cover

Table 2. Distributions of event types in labeled datasets.

Event Type	Seed Dataset		Final Dataset	
Intention	23	(7.67%)	263	(8.77%)
Action	32	(10.67%)	422	(14.07%)
Behavior	101	(33.67%)	741	(24.70%)
Reaction	37	(12.33%)	531	(17.70%)
Context	58	(19.33%)	472	(15.73%)
Nontarget	49	(16.33%)	571	(19.03%)
Total	300		3,000	

a preceding event and a following event, respectively, as the context of an event. We convert these two text segments into vectors using USE, and concatenate them to the vector of an event. The classification models are then applied to the concatenated vector.

We address $\mathbf{RQ_{extract}}$ by reporting the performance of the event type classifiers. We choose the classifier that yields the highest accuracy to determine the types of all extracted events. Reviews that contain at least one target event are referred to as *target* reviews.

3.3 Event Sequencing

Of the target reviews, many describe only one event, typically a bad app behavior in negative reviews. These "simple" stories can be collected directly for analysis. However, most target reviews are "complex" and include multiple events. Although it is natural for storytellers to describe events in the order in which they happen, such *occurrence order* is not always the same as the order in which events appear in text, i.e., *discourse order*. Additionally, it is not uncommon for a reviewer to describe multiple stories in a single review. Example 2 shows a review (for the Snapchat app) of higher complexity. Target events and their types are marked.

Example 2

★☆☆☆☆ username2, 06/10/2014

HATING SO MUCH LATELY!

I HATE how in iphones you can not zoom in to record a video$_{\text{Behavior}}$. If you zoom in and try to record$_{\text{Action}}$ it goes back to normal$_{\text{Behavior}}$. How ANNOYING! I also HATE how when someone sends me a conversation$_{\text{Action}}$ my music will stop playing$_{\text{Behavior}}$ because I opened what they sent me$_{\text{Action}}$. It's not a snap necessarily$_{\text{Context}}$ it's a simple conversation$_{\text{Context}}$. Also my snapchat sometimes says like memory full$_{\text{Behavior}}$ when I try to take or record a snapchat$_{\text{Action}}$. It's so ANNOYING.

This review provides three short stories, each describing a bug in the app, and the discourse orders in the last two stories are different from their occurrence orders. For example, in the second story, the order in which the events occurred was: ⟨someone sends me a conversation → I opened what they sent me → my music will stop playing⟩. In this step, we seek to determine whether two events are from the same story, and, if so, what their occurrence order is. Thus, we can combine related events into stories in their occurrence order.

We assume that events in the same sentence belong to the same story, and sentences that are separated by k ($k \geq 3$) Nontarget events are from different stories. We adopt heuristics based on language cues to determine the order of events from the same sentence. If there is no listed

language cue between event phrases from the same sentence, we order them as they appear in the text. We include two additional heuristics for sentences that are adjacent to each other. Table 3 shows all the heuristics we adopt in this study (→ indicates the occurrence order between events, and [SEP] marks the sentence boundary).

Table 3. Heuristics to determine event relations.

Event Order	Sentence Structure
$e_1 \rightarrow e_2$	e_1, *before / until / then* e_2 e_1 [SEP] *And then* e_2
$e_2 \rightarrow e_1$	e_1, *after / when / whenever / every time / as soon as* e_2 e_1, *if / because* e_2
Separate	e_1 [SEP] *Also / Additionally* e_2

Event relation classification. These heuristics cover only a portion of the extracted events. For the remainder, we frame the event sequencing problem as a three-way classification problem on two event phrases: given a pair of events, e_1 and e_2, where e_2 appears after e_1 in the text, does e_2 occur before e_1, after e_1, or in a separate story from e_1? We train a model on this event relation classification task. We apply this model, combined with heuristics, to determine the relation between every event and its preceding event in the text. In this fashion, we can sequence all events in a review into stories. To form a training set for such a classifier, we glean related and unrelated event pairs using the heuristics listed in Table 3, with the language cues removed. The classifier determines event relations based solely on the meanings of the events.

We experiment with different classification methods as shown below. We report and compare their performance. The model with the highest accuracy is to be chosen to sequence all extracted event phrases. The extracted stories, i.e., sequences of ordered events, are to be used for the following component of Saga.

SVM / MLP. Since this task has two inputs, we first encode the event phrases into vectors, and then concatenate them into a large vector as the input to a classifier. For encoding the event phrases, we experiment with USE, average GloVe [35] vector, and average Word2Vec [30] vector.

LSTM. We convert each word in the two events into a vector using GloVe or Word2Vec, and employ a sequential model, Long Short-Term Memory (LSTM) [18] network, for the classification.

Fine-tuned BERT. BERT [7] is a transformer-based NLP technique that supports the input being two sentences. We fine-tuned a pre-trained BERT model[5] for event relation classification.

We address $\mathbf{RQ_{relate}}$ by reporting Saga's accuracy on event relation classification.

3.4 Story Collection

The foregoing steps show how Saga extracts stories, i.e., sequences of events, and their structures as patterns of event types. We now describe how Saga collects stories based on their structures. In this component, we make the following assumptions:

[5]https://huggingface.co/bert-base-uncased

Nontarget events do not contribute to the structure of a story, except for being separators from other stories;

Context events can appear in any part of a story, so where they appear does not affect the structure of the story;

Adjacent events of the same type are potentially a coherent chain of events and can be considered collectively.

As our main focus is user-app interaction stories, these assumptions are reasonable and help simplify our analysis. Nontarget events are often related to user opinions or emotions, which are not the focus of our study. Context events contribute to the understanding of the stories, but their position in the story may not be pivotal. We include these events in our helpfulness study, as they are useful to software developers. As adjacent events of the same type, e.g., multiple user actions in a row, describe a coherent flow of occurrences, differentiating the exact numbers of repetitions may not be constructive in understanding story structures. Based on these assumptions, we ignore Nontarget and Context events in the collection process. We mark the same event type occurring in adjacent events with a plus sign. Thus, the stories in Examples 1 and 2 can be represented by the patterns in Table 4. Here, A+B represents a structure of multiple user action events followed by an app behavior event.

Table 4. Story structures in the examples.

Story	Review	Structure
s_1	Example 1	IABR+B
s_2	Example 2	BAB
s_3	Example 2	A+B
s_4	Example 2	AB

After converting the story structures into such patterns, we can search for stories based on them. For example, if we search the pattern ⟨Action → Behavior⟩ to gather stories about app problems caused by user actions, we would collect all four stories. If we search the pattern ⟨Behavior → Reaction⟩ to understand users' reactions to app behaviors, only story s_1 would be retrieved.

We count the number of stories with each distinct structure and present the most common ones. In our counting, if the story structure contains a repetition of event types, it is counted twice, both as a structure with and without repetition. For example, a story with the structure A+B is counted toward both the structure A+B and AB. AB structure is the most common structure in Table 4, appearing in two stories.

Each story structure contains one or more substructures. We investigate the most common substructures. We treat the collection of frequent story substructures as a sequential pattern mining problem. We adopt the Generalized Sequential Pattern (GSP) [41] algorithm to mine frequent story patterns. Like our structure counting process, our mining process differs from the standard GSP only in that the repetition of event types is counted twice. The pattern AB appears in all four stories in Table 4, and is extracted as a frequent pattern.

We address $\mathbf{RQ_{collect}}$ by presenting the most common story structures and substructures discovered from the app review dataset.

3.5 An Empirical Study

To show that analyzing stories based on their structures are helpful to software developers, we conducted a small-scale human verification. Leveraging Saga, we collected 200 stories from app

reviews for the Snapchat app with different event patterns for searching. We retrieved 25 stories for each of the following target patterns, A+B+, C+B+, B+R+, and I+A+. We then randomly selected 100 stories as a control group. We employed five graduate students majoring in Computer Science who are familiar with Snapchat and software development. We divided the stories among the annotators, such that each story was investigated by two people. We asked the annotators to rate each story in terms of its helpfulness toward the understanding of app problems, user retention, and user expectation, respectively. They rated the helpfulness of a story on a Likert scale, where 5 means very helpful, and 1 means not helpful at all. We compute the average helpfulness scores of each target pattern toward the three developer goals. A significant improvement of helpfulness scores over randomly selected stories would indicate the helpfulness of Saga for software developers.

4 RESULTS

We applied Saga on the 2,118,942 negative reviews, as described in Section 3.1. We address $\mathbf{RQ_{extract}}$, $\mathbf{RQ_{relate}}$, and $\mathbf{RQ_{collect}}$ by reporting Saga's performance and results in its three parts.

4.1 Event Identification

We applied the event extraction process on all reviews and extracted 9,305,505 event phrases. We answer $\mathbf{RQ_{extract}}$ by reporting the performance of event type classification. We discuss the quality of event extraction in Section 5.

As we mentioned in Section 3, we considered KNN, SVM, DT, and MLP, both with ("context") and without ("event") the context information for the classification of event types. We compare their performance via 10-fold cross-validation. We adopted the implementations of scikit-learn.[6] For each model, we experimented with different parameters and haved reported only the results with the best parameters. In the KNN model, $k = 7$. We adopted the radial basis function (RBF) kernel for SVM. For decision trees, the maximum depth was seven. The intermediate layer size of $\text{MLP}_{\text{event}}$ was 55 and that of $\text{MLP}_{\text{context}}$ was 96.

Additionally, we report the precision and recall of event identification by considering target events as relevant, and NONTARGET events as irrelevant. Table 5 shows the performance of each classification model.

Table 5. Performance of event type classification.

Model	Accuracy (6-class)	Precision	Recall	F-1 Score
$\text{KNN}_{\text{event}}$	0.661	0.922	0.951	0.936
$\text{SVM}_{\text{event}}$	0.741	0.956	0.932	0.944
DT_{event}	0.539	0.896	0.906	0.901
$\text{MLP}_{\text{event}}$	0.717	0.953	0.930	0.942
$\text{KNN}_{\text{context}}$	0.584	0.888	0.954	0.920
$\text{SVM}_{\text{context}}$	0.718	0.955	0.914	0.934
$\text{DT}_{\text{context}}$	0.529	0.894	0.910	0.902
$\text{MLP}_{\text{context}}$	0.720	0.949	0.932	0.941

The results show that the two SVM models and the two MLP models yield comparable results, and perform well on the identification of target events. The context information does not bring notable

[6] https://scikit-learn.org/stable/

difference in the classification. We choose the model that yields the highest six-class accuracy, SVM_{event}, to classify all the extracted events for the following steps. The distribution of event types in the entire dataset is shown in Table 6.

Table 6. Distributions of event types overall and in simple reviews.

Event Type	Event Count	Simple Reviews
Intention	203,053 (2.18%)	16,256 (3.42%)
Action	658,931 (7.08%)	31,377 (6.60%)
Behavior	3,065,360 (32.94%)	334,549 (70.37%)
Reaction	591,624 (6.36%)	35,507 (7.47%)
Context	1,201,579 (12.91%)	57,756 (12.15%)
Nontarget	3,584,958 (38.53%)	–
Total	9,305,505	475,445

Of all target reviews, 373,470 (17.63%) contain no event or only Nontarget events. These reviews mainly express a reviewer's opinions, requests, ratings, or other information. 475,445 (22.44%) reviews contain only one target event. Table 6 shows the distribution of event types in these simple reviews. The next two components of Saga consider only the remaining 1,270,027 (59.94%) complex reviews that contain multiple target events, of which 733,748 (34.63%) contain at least two types of events.

4.2 Event Sequencing

Based on the method mentioned in Section 3.3, we determined the order of 1,005,166 event pairs by the heuristics, 32.4% of all event pairs. From these event pairs, we randomly sampled 20,000 event pairs for each relation type, $e_1 \rightarrow e_2$, $e_2 \rightarrow e_1$, and Separate. For event relation classification, we divided these 60,000 event pairs into a training set (90%) and a testing set (10%) and compared the performance of different classification models. The event relation classification performance of different models is shown in Table 7.

Table 7. Performance of event relation classification.

Model	Accuracy	Model	Accuracy	Model	Accuracy
SVM_{GloVe}	0.737	MLP_{GloVe}	0.727	$LSTM_{GloVe}$	0.722
$SVM_{Word2Vec}$	0.728	$MLP_{Word2Vec}$	0.718	$LSTM_{Word2Vec}$	0.714
SVM_{USE}	0.752	MLP_{USE}	0.736	$BERT_{base}$	0.797

The fine-tuned BERT yielded the highest accuracy. Using both heuristics and this classification model, we obtained 2,500,580 stories from the 1,270,027 complex reviews from the previous step.

4.3 Story Collection

In this step, we only consider Intention, Action, Behavior, and Reaction events, and ignore other events. Of the 2,500,580 stories from the previous step, 269,409 (10.8%) contain only Context events. When we ignore Context events, 1,558,156 (62.3%) stories are composed of only one type of event out of the four remaining types. The rest 673,015 (26.9%) stories are complex stories with

Table 8. Common story structures.

Simple Stories		Complex Stories (freq > 1%)			
Length 1		Length 2		Length 3	
B	855,630 (54.9%)	AB	176,661 (26.25%)	BAB	39,291 (5.84%)
B+	365,361 (23.4%)	BR	85,807 (12.75%)	BRB	19,794 (2.94%)
R	152,259 (9.77%)	BA	60,310 (8.96%)	ABR	13,030 (1.94%)
A	88,178 (5.66%)	RB	56,928 (8.46%)	ABA	9,431 (1.40%)
I	55,613 (3.57%)	AB+	52,817 (7.85%)	BAB+	7,783 (1.16%)
R+	25,592 (1.64%)	IB	34,629 (5.15%)		
A+	12,747 (0.82%)	B+R	20,414 (3.03%)		
I+	2,776 (0.18%)	BI	16,091 (2.39%)		
		B+A	12,858 (1.91%)		
		AR	12,486 (1.86%)		
		RB+	9,943 (1.48%)		
		A+B	9,815 (1.46%)		
		IB+	8,424 (1.25%)	Length 4	
		RA	7,793 (1.16%)		
		R+B	7,249 (1.08%)		
		BR+	7,075 (1.05%)	ABAB	8,869 (1.32%)

more than one type of event. Table 8 shows the story structures most common in simple and complex stories. We note that 22 structures are common (frequency of more than 1%) in complex stories.

Most simple stories (78.3%) in app reviews describe only the apps' behaviors. Such simple behavior stories constitute 48.8% of all 2,500,580 stories in complex reviews. The most common length-two structure is ⟨Action → Behavior⟩ for complex stories, in which an app behavior is caused or triggered by a user action.

To obtain the most common story substructures, we ran Generalized Sequential Pattern (GSP) on the complex stories only. We retained 43 frequent substructures that appeared in more than 1% (6,730) of the complex stories, as shown in Table 9. The most frequent type of event is app Behavior, followed by user Action and Reaction, which is not surprising since we have targeted negative reviews. Most described app behaviors are problems, and reviewers typically describe what they did before and after encountering the problems.

Many frequent substructures are interpretable. For example, the substructure BRB potentially describes a part of a story in which the app is exhibiting a problem, the user tries to fix it, but the problem persists. The substructure ABR likely describes a part of a story in which a user's action causes an app behavior, and the user reacts to this behavior. We list some example stories (with minor edits) that contain frequent substructures in Table 10. Note that the patterns are based on the occurrence orders of events, not their discourse order in the text.

4.4 Manual Verification

We showed the 200 sample stories to our annotators, and asked them to rate the stories in terms of their helpfulness toward the developer goals of understanding app problems, user retention, and user expectation. Our annotators achieved moderate agreements for all three goals (Cohen's kappa

Table 9. Frequent substructures appearing in > 1% of the complex stories.

Length 1		Length 2		Length 3	
B	629,562 (93.54%)	AB	294,096 (43.70%)	BAB	67,178 (9.98%)
A	422,417 (62.76%)	BR	161,000 (23.92%)	BRB	34,883 (5.18%)
R	285,226 (42.38%)	BA	157,842 (23.45%)	ABA	30,222 (4.49%)
B+	193,518 (28.75%)	RB	115,699 (17.19%)	ABR	28,600 (4.25%)
I	117,656 (17.48%)	AB+	85,970 (12.77%)	B+AB	14,836 (2.20%)
A+	41,255 (6.13%)	IB	59,579 (8.85%)	BAB+	13,618 (2.02%)
R+	37,069 (5.51%)	B+R	44,761 (6.65%)	RBR	13,261 (1.97%)
		B+A	39,033 (5.80%)	BAR	12,690 (1.89%)
		BI	37,440 (5.56%)	ARB	10,759 (1.60%)
		AR	37,371 (5.55%)	BIB	10,595 (1.57%)
		A+B	28,471 (4.23%)	RAB	10,536 (1.57%)
		RA	28,092 (4.17%)	BRA	9,424 (1.40%)
		RB+	21,953 (3.26%)	AB+R	8,068 (1.20%)
Length 4		R+B	16,642 (2.47%)	B+RB	8,050 (1.20%)
ABAB	14,760 (2.19%)	IA	15,670 (2.33%)	RBA	7,719 (1.15%)
ABRB	7,162 (1.06%)	IB+	14,580 (2.17%)	AB+A	7,429 (1.10%)
BABR	7,025 (1.04%)	BR+	14,382 (2.14%)		
BABA	6,743 (1.00%)	AI	13,530 (2.01%)		
		IR	13,178 (1.96%)		
		BA+	11,381 (1.69%)		
		A+B+	9,182 (1.36%)		
		B+I	8,902 (1.32%)		
		RI	7,525 (1.12%)		

is 0.441, 0.541, and 0.455 for app problems, user retention, and user expectation, respectively). We calculated the average score of each pair of annotations as the final score.

Table 11 shows the average score of stories with each pattern, as well as random stories. Here, simple problem stories are the stories with B events but no A, C, or R events. We can see that stories with specific substructures received significantly higher helpfulness scores than random stories or stories that only describe app problems.

These results make intuitive sense. From a developer's point of view, stories with B events describe app behaviors that users experience. In negative reviews, they typically indicate app problems. With the combination of A, C, or R events, the stories provide more details on the scenarios where the problems happened. Stories with B+R+ patterns describe users reactions to unexpected app behaviors, often including information on why the users leave the apps. I+A+ stories only describe users intentions and actions, which are not directly related to app failures. However, they are, similar to B+R+ stories, part of the confluence of users and apps, describing how users act in expectation of an app functionality. These stories will bring insights into the negotiated user-app interactions.

5 DISCUSSION AND FUTURE WORK

We presented Saga, a framework for analyzing stories present in app reviews. We now discuss the merits, threats to validity, and limitations of Saga.

Table 10. Example stories for some structures.

Struct.	Type	Events
B+	[B]	*This new format is so awful*
	[B]	*Half the time this app "can not get weather data"*
	[N]	*(When) it does*
	[B]	*it is slow to load, difficult to navigate, and unnecessarily convoluted*
AB	[A]	*(when) I'm typing to another person*
	[C]	*& they are there*
	[B]	*The yellow button doesn't always turn blue*
	[N]	*FIX IT SNAPCHAT!*
ABRB	[N]	*I love Pandora*
	[A]	*(even though) I just started listening to Pandora for the first time during the day*
	[B]	*(But often times) I'm unable to skip songs*
	[R]	*I've tried quitting and reopening…*
	[B]	*None of which work/help!!*
	[N]	*What's up with this?*
IABR	[I]	*I want to be able to delete saved chats!!!*
	[A]	*(Because if) I accidentally tap a message*
	[B]	*(then) it becomes bolded font and saves*
	[R]	*(yet) I can't unsave it!*
	[N]	*FIX IT!!!*

5.1 Merits

Saga digs into different event types and story structures in app reviews. This framework benefits researchers and practitioners in both software engineering and the computational social media community.

Event type in app reviews. Saga targets event types related to user-app interactions. Identifying events of specific types can help extract information from stories, potentially answering questions about them. For example, Behavior events in negative reviews typically describe problems that the users have experienced. And, Intention events describe the functionalities the users wish to use. Such events serve as indicators of the reviewer's expectations. Reaction events are not only illuminating for user retention, but also instrumental in the study of agency in user-app interaction. Saga extends previous work by incorporating a more diverse roster of event types, and shows that they need to be considered by researchers with different research goals.

Event sequencing. A negative review may include multiple stories, mapping to bug reports or feature requests, which should not be treated collectively. We find that, on average, each multi-event app review contains 1.41 stories, and 26.6% of such reviews contain two or more stories. Saga can identify individual stories and sequentially order the events in them, which provides a deeper understanding than aggregate notions currently seen in the analysis of stories on social media, e.g., as in [14, 45].

Table 11. Average helpfulness scores of different stories toward different goals (p_s denotes p-value against simple problem stories; p_r denotes p-value against random stories).

Goal	Simple Problem Stories	Random	Stories w/ Pattern	Score	p_s	p_r
App Problem	3.578	3.435	A+B+	4.163	0.003	0.000
			C+B+	4.118	0.009	0.000
			B+R+	4.136	0.005	0.000
			I+A+	3.900	-	-
User Retention	1.689	1.825	A+B+	1.596	-	-
			C+B+	1.735	-	-
			B+R+	2.652	0.001	0.005
			I+A+	1.617	-	-
User Expectation	3.467	3.275	A+B+	3.125	-	-
			C+B+	3.039	-	-
			B+R+	2.288	-	-
			I+A+	4.133	-	0.000

Story structures. Saga introduces a novel way of analyzing stories based on their semantic structure. From a software developer's point of view, Saga provides a different perspective when retrieving useful information from user reviews. We have seen that the structure of a story (or set of stories) in an app review is a good indicator of the overall message the author wishes to convey, and stories of specific substructures prove more helpful than average for supporting associated information goals. Although we verify this from the developers' point of view in our specific setting, we posit that authors naturally adopt certain narrative conventions (in small subcommunities) when trying to convey specific types of opinions. Developers can take advantage of these conventions and achieve their data mining objectives.

Saga also provides a way of analyzing writers' story-writing behaviors in the app review setting. The way that an author brings up various event types may be coupled with deeper implications than the type of feedback to an app. For example, some review writers describe multiple stories in one review, while others may only include one simple story, or just some **Nontarget** events that express their opinions or ratings. The difference may be indicative of the users' usage time with the app, their sentiment of the interaction, their willingness of providing useful feedback, and user's expectations of writing the reviews.

Additionally, the results of our story structures collection can shed light into the study of user-app interaction within a social structure. For example, the overwhelming majority of the simple stories in app reviews are about the app's (unexpected) behaviors. This result means that, when users write short stories / short negative reviews, they mostly focus on describing the problems of the apps as the most useful feedback. Based on our limited observations, app users are less likely to include similar stories in positive reviews. Instead, they tend to include more praises and requests for future improvements. Following Saga, researchers may dig deeper in terms of social behaviors in review writing. However, we leave those to future work.

5.2 Threats to Validity

We targeted negative reviews, which are richer in stories with actions in them. Intuitively, it makes sense that reviewers would do not report an app's behavior if their expectations are met. Our method may not generalize well to stories in neutral or positive reviews, which might involve events outside the direct scope of an app, as in "Snapchat helped me find my life partner."

We trained our event relation classifier with related event pairs identified with heuristics. These heuristics select event pairs that occur in the same sentence or adjacent sentences with explicit cues. A classifier trained on such event pairs may not generalize well on event pairs extracted from different sentences.

5.3 Limitations and Future Work

We identify the following limitations in Saga as well as potential future work to address them.

App Reviews. In this work, we target mobile app reviews as a source for user-generated stories. App stores are not intended as social media, and stories shared there may not be representative of stories shared on other social networks. However, stories in app reviews are meant to be shared nevertheless, either to app developers or among current or prospective app users. Our research can spark a rich line of studies into social behaviors in the mobile app field. We leave the investigation of stories in other social media to future work.

Target event types. Saga targets five types of events. However, events in some stories may be more diverse. For example, some stories (e.g., in a review of Uber) may describe actions of other users, e.g., Uber drivers, that affected the storyteller. Such information may not be part of a story but is valuable nevertheless in terms of the goals that the author is trying to achieve. Additionally, a lot of the NONTARGET events in this study describe user opinions. One may argue that these events also affect the progression of the story. Additionally, Saga is limited to event types specific to app reviews. For the analysis of stories in other online social platforms, event types may be more diverse and may not be easily categorized.

Text quality. Typos, grammatical errors, and incorrect punctuation hurt the performance of the NLP tools we rely on, including POS taggers and dependency parsers. Some reviews contain expressions that are difficult to handle. For example, Saga could not handle a review that repeats the word *LAG* more than a hundred times. Future work includes the investigation of robust methods.

Classification performance. Although Saga yields acceptable accuracy for six-way classification, there is room for improvement. Moreover, certain event types are difficult to distinguish, e.g., as attempts. For example, a user INTENTION, e.g., *I tried to deposit a check*, can syntactically resemble a user ACTION, e.g., *I tried to take a picture*, or a user REACTION, e.g., *I tried to reinstall the app*. Contextual information could potentially help in such cases.

Implications of story structures. In this work, we focus on the analysis of story structures from a software developer's point of view. We have shown that stories with certain structures are helpful toward software improvement. However, we have not investigated what story structures are the most helpful, and what other story structures should also be considered.

Story structures may have more implications from the stand points of human behavior or social structure. For example, certain stories may indicate the users are dissatisfied with an application. The types of stories may correlate to the writer's expected outcome of providing useful feedback. User agency during the review writing process may be affected by the existing social conventions of other app users. Another challenge is to examine stories from the perspective of trust, e.g., whether

they indicate ability, benevolence, and integrity [28] on the part of the app developers. This model has been applied to software artifacts such as AI [40]. We leave such research to future work.

5.4 Broader Impact and Ethical Considerations

We collected publicly available app reviews from Apple App Store, and all of our data and models are based on this public information. We were not able to collect consent from the review writers. All identifiers, such as user IDs and names, are excluded from our study, both for annotations and analysis. Even though some of the targeted reviews included specific stories, our annotators were only shown a few event phrases, which would not be enough to identify the writers. Our study focuses on the event types and story structures, and does not rely on the identify of the storytellers. The chance of our results being misused is minimum.

Negative reviews include a large amount of emotion impressions, the extreme cases of which may not be comfortable to label. However, such cases are rare. We limited the amount of nontarget events during the crowdsourcing projects, and made sure such cases were nonexistent in the set of events to be labeled.

6 CONCLUSIONS

Saga addresses the problem of identifying stories in an under-studied social media source, namely, app reviews. A benefit of app reviews is that, although they involve stories of rich event structures, they are generally light on the complexities of human-human interactions and the concomitant psychosocial attributes of human relationships. Thus, these stories challenge our ability to extract and structure events. Interestingly, Saga can help serve information goals pertaining to app reviews based on the structures and substructures it discovers.

Saga is able to go beyond existing story understanding research in social media, which (as far as we know) is limited to analyzing general themes in narratives [2, 17] or their aggregate properties [43] but lacks the ability to tackle the stories individually and in a semantically rich manner.

A natural extension of Saga would be to enrich it with higher-level affective considerations, such as of emotion and morality that are common in narratives [46]. Such aspects may arise in positive app reviews indicating user satisfaction and negative app reviews indicating frustration where the user perceives the app was unfairly sold.

ACKNOWLEDGMENTS

Thanks to the anonymous reviewers for helpful comments. Thanks to the Department of Defense for partial support for this research under the Science of Security Lablet (Project H98230-17-D-0080).

REFERENCES

[1] Brandon Beamer and Roxana Girju. 2009. Using a Bigram Event Model to Predict Causal Potential. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*. Springer Verlag, Mexico City, Mexico, 430–441. https://doi.org/10.1007/978-3-642-00382-0_35

[2] Mack Blackburn, Ning Yu, Alex Memory, and W. Graham Mueller. 2020. Detecting and Annotating Narratives in Social Media: A Vision Paper. In *Workshop Proceedings of the 14th International AAAI Conference on Web and Social Media*. AAAI Press, Atlanta, 1–2. https://doi.org/10.36190/2020.23

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175 (2018), 1–7. http://arxiv.org/abs/1803.11175

[4] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C. Gall. 2017. Analyzing Reviews and Code of Mobile Apps for Better Release Planning. In *Proceedings of 24th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE Computer Society, Klagenfurt, Austria, 91–102. https://doi.org/10.1109/SANER.2017.7884612

[5] Bernard Comrie. 1986. *Tense*. Cambridge University Press, Cambridge, United Kingdom.

[6] Marie-Catherine de Marneffe and Christopher D. Manning. 2016. Stanford Typed Dependencies Manual. https://nlp.stanford.edu/software/dependencies_manual.pdf. [Online; originally published in September 2008; accessed: 2024-12-19].

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/V1/N19-1423

[8] Andrea Di Sorbo, Giovanni Grano, Corrado Aaron Visaggio, and Sebastiano Panichella. 2020. Investigating the Criticality of User-reported Issues through Their Relations with App Rating. *Journal of Software: Evolution and Process* 33, 3 (Sept. 2020), e2316. https://doi.org/10.1002/SMR.2316

[9] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. What Would Users Change in My App? Summarizing App Reviews for Recommending Software Changes. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. ACM, Seattle, WA, USA, 499–510. https://doi.org/10.1145/2950290.2950299

[10] Jacek Dąbrowski, Emmanuel Letier, Anna Perini, and Angelo Susi. 2019. Finding and Analyzing App Reviews Related to Specific Features: A Research Preview. In *Proceedings of International Working Conference on Requirements Engineering: Foundation for Software Quality (Lecture Notes in Computer Science, Vol. 11412)*, Eric Knauss and Michael Goedicke (Eds.). Springer International Publishing, Essen, Germany, 183–189. https://doi.org/10.1007/978-3-030-15538-4_14

[11] Jacek Dąbrowski, Emmanuel Letier, Anna Perini, and Angelo Susi. 2022. Analysing App Reviews for Software Engineering: A Systematic Literature Review. *Empirical Software Engineering* 27, 43 (March 2022), 1–63. https://doi.org/10.1007/S10664-021-10065-7

[12] Rino Falcone and Alessandro Sapienza. 2020. Trust and Autonomy for Regulating the Users' Acceptance of IoT Technologies. *Intelligenza Artificiale* 14, 1 (2020), 45–58. https://doi.org/10.3233/IA-190041

[13] Vaibhav Garg, Hui Guo, Nirav Ajmeri, Saikath Bhattacharya, and Munindar P. Singh. 2025. Understanding Mobile App Reviews to Guide Misuse Audits. *Communications of the ACM (CACM)* 0, 0 (2025), 16 pages. arXiv:2303.10795 To appear.

[14] Salvatore Giorgi, Ke Zhao, Alexander H. Feng, and Lara J. Martin. 2023. Author as Character and Narrator: Deconstructing Personal Narratives from the r/amitheasshole Reddit Community. In *Proceedings of the 17th International AAAI Conference on Web and Social Media*. AAAI Press, Limassol, Cyprus, 233–244. https://doi.org/10.1609/ICWSM.V17I1.22141

[15] Rob Grace, Kenyan Burnham, and Hyeong Suk Na. 2023. How Much Context Do Users Provide in App Reviews? Implications for Requirements Elicitation. In *Information for a Better World: Normality, Virtuality, Physicality, Inclusivity*, Vol. 13972. Springer Nature Switzerland, Cham, Switzerland, 16–25. https://doi.org/10.1007/978-3-031-28032-0_2

[16] Hui Guo and Munindar P. Singh. 2020. Caspar: Extracting and Synthesizing User Stories of Problems from App Reviews. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE)*. Association for Computing Machinery, New York, NY, USA, 628–640. https://doi.org/10.1145/3377811.3380924

[17] Hussam Habib and Rishab Nithyanand. 2023. The Morbid Realities of Social Media: An Investigation into the Narratives Shared by the Deceased Victims of COVID-19. In *Proceedings of the 17th International AAAI Conference on Web and Social Media*. AAAI Press, Limassol, Cyprus, 303–314. https://doi.org/10.1609/ICWSM.V17I1.22147

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

[19] Zhichao Hu, Elahe Rahimtoroghi, and Marilyn Walker. 2017. Inference of Fine-Grained Event Causality from Blogs and Films. In *Proceedings of the Events and Stories in the News Workshop*. Association for Computational Linguistics, Vancouver, Canada, 52–58. https://doi.org/10.18653/V1/W17-2708

[20] Zhizhao Hu and Marilyn A. Walker. 2017. Inferring Narrative Causality between Event Pairs in Films. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, Saarbrücken, Germany, 342–351. https://doi.org/10.18653/V1/W17-5540

[21] Nishant Jha and Anas Mahmoud. 2019. Mining Non-functional Requirements from App Store Reviews. *Empirical Software Engineering* 24, 6 (Dec. 2019), 3659–3695. https://doi.org/10.1007/S10664-019-09716-7

[22] Hyunjin Kang and Chen Lou. 2022. AI Agency vs. Human Agency: Understanding Human-AI Interactions on TikTok and their Implications for User Engagement. *Journal of Computer-Mediated Communication* 27, 5 (Aug. 2022), 1–13. https://doi.org/10.1093/JCMC/ZMAC014

[23] Anang Kunaefi and Masayoshi Aritsugi. 2020. Characterizing User Decision based on Argumentative Reviews. In *IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. IEEE, Leicester, UK, 161–170. https://doi.org/10.1109/BDCAT50828.2020.00002

[24] Yuzhou Liu, Lei Liu, Liu Huaxiao, and Shanquan Gao. 2020. Combining Goal Model with Reviews for Supporting the Evolution of Apps. *IET Software* 14, 1 (Feb. 2020), 39–49. https://doi.org/10.1049/IET-SEN.2018.5192

[25] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. 2016. On the Automatic Classification of App Reviews. *Requirements Engineering* 21, 3 (Sept. 2016), 311–331. https://doi.org/10.1007/S00766-016-0251-9

[26] Walid Maalej and Hadeer Nabil. 2015. Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE)*. IEEE Press, Ottawa, ON, Canada, 116–125. https://doi.org/10.1109/RE.2015.7320414

[27] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine Learning of Temporal Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, 753–760. https://doi.org/10.3115/1220175.1220270

[28] Roger C. Mayer, James H. Davis, and F. David Schoorman. 1995. An Integrative Model of Organizational Trust. *The Academy of Management Review* 20, 3 (July 1995), 709–734. https://doi.org/10.5465/amr.1995.9508080335

[29] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. 2016. Analyzing and Automatically Labelling the Types of User Issues That Are Raised in Mobile App Reviews. *Empirical Software Engineering* 21, 3 (June 2016), 1067–1106. https://doi.org/10.1007/S10664-015-9375-7

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS)*. Neural Information Processing Systems Foundation, Lake Tahoe, Nevada, 3111–3119.

[31] Seyed Abolghasem Mirroshandel and Gholamreza Ghassem-Sani. 2012. Towards Unsupervised Learning of Temporal Relations between Events. *Journal of Artificial Intelligence Research* 45, 1 (Sept. 2012), 125–163. https://doi.org/10.1613/JAIR.3693

[32] Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018. Joint Reasoning for Temporal and Causal Relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2278–2288. https://doi.org/10.18653/V1/P18-1212

[33] Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018. Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 841–851. https://doi.org/10.18653/V1/N18-1077

[34] Dragoş M. Obreja. 2024. When Stories Turn Institutional: How TikTok Users Legitimate the Algorithmic Sensemaking. *Social Media + Society* 10, 1 (Jan. 2024), 1–11. https://doi.org/10.1177/20563051231224114

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/V1/D14-1162

[36] Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2Mind: Commonsense Inference on Events, Intents, and Reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computer Linguistics, Melbourne, Australia, 463–473. https://doi.org/10.18653/V1/P18-1043

[37] Beatrice Santorini. 1995. *Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd printing)*. Technical Report. Department of Computer and Information Science, University of Pennsylvania.

[38] Maarten Sap, Ronan J. LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, Honolulu, Hawaii, USA, 3027–3035. https://doi.org/10.1609/AAAI.V33I01.33013027

[39] Ignacio Siles, Luciana Valerio-Alfaro, and Ariana Meléndez-Moran. 2022. Learning to like TikTok … and not: Algorithm awareness as process. *New Media & Society* 26, 10 (Dec. 2022), 5702–5718. https://doi.org/10.1177/14614448221138973

[40] Amika M. Singh and Munindar P. Singh. 2023. Wasabi: A Conceptual Model for Trustworthy Artificial Intelligence. *IEEE Computer* 56, 2 (Feb. 2023), 20–28. https://doi.org/10.1109/MC.2022.3212022

[41] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology*, Peter Apers, Mokrane Bouzeghoub, and Georges Gardarin (Eds.), Vol. 1057. Springer Berlin Heidelberg, Avignon, France, 3–17. https://doi.org/10.1007/BFB0014140

[42] Kamonphop Srisopha, Chukiat Phonsom, Mingzhe Li, Daniel Link, and Barry Boehm. 2020. On Building an Automatic Identification of Country-Specific Feature Requests in Mobile App Reviews: Possibilities and Challenges. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 494–498. https://doi.org/10.1145/3387940.3391492

[43] Lu Sun, F. Maxwell Harper, Chia-Jung Lee, Vanessa Murdock, and Barbara Poblete. 2023. Characterizing and Identifying Socially Shared Self-Descriptions in Product Reviews. In *Proceedings of the 17th International AAAI Conference on Web and Social Media*. AAAI Press, Limassol, Cyprus, 808–819. https://doi.org/10.1609/ICWSM.V17I1.22190

[44] Andrew Truelove, Farah Naz Chowdhury, Omprakash Gnawali, and Mohammad Amin Alipour. 2019. Topics of Concern: Identifying User Issues in Reviews of IoT Apps and Devices. In *Proceeding of the 1st IEEE/ACM International Workshop on Software Engineering Research Practices for the Internet of Things (SERP4IoT)*. IEEE, Montréal, Québec, Canada, 33–40. https://doi.org/10.1109/SERP4IOT.2019.00013

[45] Ruijie Xi and Munindar P. Singh. 2024. The Blame Game: Understanding Blame Assignment in Social Media. *IEEE Transactions on Computational Social Systems (TCSS)* 11, 2 (April 2024), 2267–2276. https://doi.org/10.1109/TCSS.2023.3261242

[46] Ruijie Xi and Munindar P. Singh. 2024. Morality in the Mundane: Categorizing Moral Reasoning in Real-life Social Situations. In *Proceedings of the 18th International AAAI Conference on Web and Social Media (ICWSM)*. AAAI Press, Buffalo, New York, 1648–1660. https://doi.org/10.1609/icwsm.v18i1.31415

[47] Yu Zhou, Yanqi Su, Taolue Chen, Zhiqiu Huang, Harald Gall, and Sebastiano Panichella. 2021. User Review-Based Change File Localization for Mobile Applications. *IEEE Transactions on Software Engineering* 47, 12 (Jan. 2021), 2755–2770. https://doi.org/10.1109/TSE.2020.2967383