ABSTRACT

GUO, HUI. Extraction and Inference of Event Pairs in Stories. (Under the direction of Dr. Munindar P. Singh.)

A story, in the sense of natural language processing, comprises a sequence of events. The relations between pairs of events are the building blocks of a coherent story. We consider the *inferential relations* between events. Two events are *inferentially related* if their sequential order can be inferred by commonsense. In certain types of texts, such as app reviews and breach reports, inferentially relations event pairs bear particular importance as they can be considered as the take-away messages or lessons learned. Their extraction should be deliberately investigated. Inference on event pairs and stories can help the understanding and reasoning of stories in important documents. This research investigates the extraction and inference of event pairs, in the form of event phrases, that are inferentially ordered. We address the importance of extracting such events pairs, how their orders can be inferred, and how their inferred orders can facilitate event inference in stories.

This research (1) develops CASPAR, a method for extracting and analyzing user-reported event pairs regarding app problems from app reviews, and (2) proposes EPOCI, the task of event pair ordering based on commonsense inference, and its solutions. CASPAR collects event pairs that describe (1) user actions and (2) app problems that are triggered by the user actions. These event pairs capture the essence of the bug-report type of app reviews, and provide information that facilitates developers to maintain and improve the app's functionality and user experience. To target the EPOCI task, we (1) collect a dataset of event pairs ordered by commonsense inference (the EPOCI dataset) through manual annotations, and (2) propose deep learning methods for the classification of event pairs based on their inferential orders. We report the performance of the proposed methods and necessitate future improvement of the solutions.

CASPAR discovers high-quality event pairs regarding app problems from app reviews. By presenting CASPAR, we show that inferentially ordered event pairs capture important information of stories. Our proposed methods solve the EPOCI task with moderate accuracy. We posit that information learned from event pairs can be leveraged for event inference in stories with longer sequences of events, which motivates our research plan.

Extraction and Inference of Event Pairs in Stories

by
Hui Guo

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2019

APPROVED BY:

Dr. Christopher Healey	Dr. Arnav Jhala
Dr. Collin Lynch	Dr. Munindar P. Singh Chair of Advisory Committee

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Munindar P. Singh, for his help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

The relations between event pairs are the links that connect a story. Previous research on event pair relations have been focused on the manners of their manifestation in texts, such as the frequencies of the orders in which they occur. However, the occurrence of an event, in itself, often presumes certain facts and yields plausible implications. A human with commonsense can connect these presumptions and implications, and a direct result of having such commonsense knowledge is the ability to immediately judge the natural order of two events, should the underlying information be sufficient.

For example, *eating an apple* presumes prior possession of an apple, while *buying an apple* implies subsequently possessing an apple. Naturally, buying an apple should precede eating an apple by commonsense. We consider the *inferential relation* between two events. Two events are *inferentially ordered* if their sequential order can be inferred based on commonsense alone, assuming they are from the same coherent story, and they are therefore inferentially related.

We posit that the inferential relations between events are more informative in inferring the plausible follow-up events given the evidence event(s) than temporal or causal relations. Temporal relations between events assert how they happen in reality, regardless of their underlying connections. Causal relations between events indicate the events that are probable results of others, instead of plausible follow-up events.

One type of inferentially ordered event pairs of practical importance is the mini stories users report about user-app interactions in app reviews, in which (1) user actions, indicative of use cases or user expectations and (2) associated app behaviors, in reaction to the user actions. In negative reviews, the reported app behaviors are usually indicative of problems that violate the users' expectations. Such action-problem pairs are (1) more closely related than temporal relations, as user actions trigger the app problems, and (2) not necessarily causally related, as user actions are usually not the causes of app problems. Being able to identify such event pairs and infer app problems would enable developers in better maintaining and improving their app's functionality and enhancing user experience.

1.1 Inferentially Related Event Pairs

Previous studies address relations between events based on how they appear in the texts. Temporal relations can be extracted using heuristics and engineered feature, and casual relations can be estimated by how much more frequently some temporal orders appear than their reverse orders. However, the inferential order of two events may not be easily inferred by their presentational features.

Consider the word *when* in a temporal clause of a sentence. Suppose an event is described in the main clause and another in the temporal clause. The *when* event can happen either before or after the main event. Take the following two sentences as an example:

Example 1

Sentence 1: I warned her to bring sunscreen *when* she went to the beach.
e_{main} = I warned her to bring sunscreen.
e_{when} = She went to the beach.
Sentence 2: She enjoyed wearing sunscreen *when* she went to the beach.
e_{main} = She enjoyed wearing sunscreen.
e_{when} = She went to the beach.

In these two sentences, the *when* events are the same. However, when we read them, we naturally think that the event in the main clause of Sentence 1 happens *before* the *when* event, whereas that of Sentence 2 happens *after* it. A human with commonsense will accept that *bringing sunscreen*, *going to the beach*,and then *wearing sunscreen* is the natural order of these events. We posit that understanding such inferential order of events is a prerequisite for reliable event inference.

In certain types of stories, the inferential order of events can often be assumed. In app reviews, users describe their interaction stories with the apps. One typical type of event pair is a action-

problem pair, which include a user action event and an app problem event triggered by it. If we can identify events that are user actions or app problems, we can extract this type of inferentially ordered event pairs with simple heuristics. The following is an example review for The Weather Channel app[1] from Apple App Store.

Example 2

★☆☆☆☆ username1, 05/29/2014

Somebody messed up!

Horrible. What on earth were these people thinking. I'm going to look for another weather app. It hesitates when I try to scroll thru cities. I'm so irritated with this fact alone that I'm not going to waste my time explaining the other issues.

The review in Example 2 contains a pair of ordered events: (1) a user action, *trying to scroll through cities* and (2) the app's problematic behavior in response, *app hesitating*. Naturally, the app problem event happens after the user action.

1.2 Research Questions

The key objectives of this research are: (1) to study the feasibility and reliability of extracting inferentially ordered event pairs of a targeted type from stories, (2) to investigate the task of automatically inferring the inferential order between events and the natural follow-up event of a given event, and (3) to explore the practicality of tackling event inference in stories based on information learned from inferentially ordered event pairs. Therefore, we identify the following research questions:

RQ$_{\text{extract}}$ How can we effectively extract targeted event pairs from stories?

RQ$_{\text{infer-pair}}$ How can we effectively conduct automatic event inference on event pairs to (1) determine their inferential order and (2) infer plausible follow-up events based on one event?

RQ$_{\text{infer-story}}$ How can we effectively conduct event inference on stories based on information learned from event pairs?

Research on the extraction of inferentially related event pairs is lacking. Certain types of inferentially related event pairs, such action-problem pairs from app review, hold valuable information. Collecting such event pairs is an important information extraction task of practical significance. We address **RQ$_{\text{extract}}$** to investigate the effective extraction process to find the targeted event pairs. This

[1] `https://apps.apple.com/us/app/weather-the-weather-channel/id295646461`

extraction task is nontrivial in that the inferential relations, unlike temporal relations, may rely on the meanings of events rather than their syntactic features.

We answer $\mathbf{RQ_{infer\text{-}pair}}$ to dive deeper to the inferential relations between events once they are extracted. Inference on event pairs is a simplified but fundamental task of inference on stories. We investigate different inference models for this task, and compare their performance. Inference on inferentially related event pairs is challenging with the lack of training data and standardized tests.

Although inference on event pairs is useful, inference in longer sequences of events bears more practical significance, since regular stories usually contain more than two events. Previous studies on event inference consider events as individual input, without considering the interplay between the event pairs. We address $\mathbf{RQ_{infer\text{-}story}}$, and tackle event inference tasks with the help of the information learned from event pairs. However, there are no existing models for this purpose. We will investigate different neural networks and compare their performance in event inference tasks.

1.3 Motivation

A story, in the sense of natural language processing, comprises a sequence of events. Stories are common in user-generated texts, such as reviews, weblogs, and reports. Understanding stories and events can benefit research on such documents, such as information retrieval and document understanding.

App users often describe their interaction stories with apps in their reviews. Application distribution platforms, such as Apple AppStore and Google Play Store, provide critical pathways for users to provide their feedback in the form of ratings and reviews [29]. User reviews have attracted much research interest of late [6, 9, 21, 23, 31]. However, current approaches focus on arguably the more superficial aspects of reviews, such as their topics and the reviewer's sentiment for an app or a feature. In contrast, we observe that reviews often carry deeper knowledge that would be valuable if it were extracted and provided to developers. Specifically, we have found that a user's review of an app often tells a mini story about how the user interacted or attempted to interact with the app. This story describes what function the user tried to bring about and how the app behaved in response. These stories not only serve as de facto deployment reports for an app, but also express users' expectations regarding the app. They contain valuable information for developers in maintaining and improving their applications.

User-reported stories in app reviews may also raise security concerns. For example, Chatterjee et al. [5] have reported that app reviews may reveal potential misuses of legit apps that enable intimate partner surveillance (IPS). We have found that some app reviews describe users' IPS activities, of which other users as well as application distribution platforms, such as Apple's App

Store, should be more aware. Studying such misuse stories in app reviews is of great practical significance in promoting the security of applications as well as the society.

In the healthcare domain, the U.S. Department of Health and Human Services (HHS) is legally required to maintain and publish a dataset of breach reports that describe incidents in which protected health information (PHI) was missing, stolen, improperly disposed of, or impermissibly accessed or disclosed [12]. In addition, they record the actions that responsible parties have taken to prevent, detect, and recover from future breaches. The knowledge that resides in such reports is valuable in refining system requirements since they contain key information regarding the prevention of and recovery from future breaches of similar kind [22, 37]. Best practices in the past described in these textual artifacts can be considered as suggestions for future practices and a great supplement to legal requirements. This knowledge can be extracted by finding the common flow of historical event occurrences. In a previous study [10], we have extracted security and privacy requirements from such natural language artifacts presented as short stories, and shown that an event inference model is effective in suggesting useful requirements for security practitioners and end users to take after a breach has happened.

1.4 Contributions and Organization

We address our research questions through three projects, which we detail in this section.

1.4.1 CASPAR: Collecting and Analyzing Stories of Problems in App Reviews

We develop CASPAR, a method for collecting and analyzing user-reported mini stories regarding app problems from app reviews. Specifically, we target the action-problem pairs in negative app reviews, which act as bug reports that would help developers in better maintaining and improving their apps' functionality and user experience.

CASPAR abstracts event pairs from stories in reviews. By extending and applying natural language processing and deep learning, CASPAR extracts ordered events from app reviews, classifies them as user actions or app problems, and conducts inference on action-problem event pairs. CASPAR also builds and trains an inference model with the extracted event pairs to predict possible app problems for different use cases.

1.4.1.1 Research Questions

CASPAR addresses the modified versions of $\mathbf{RQ_{extract}}$ and $\mathbf{RQ_{infer\text{-}pair}}$, specific to the action-problem pairs in negative app reviews.

$RQ_{extract}$ How effectively can we extract app problem stories as action-problem pairs from app reviews?

$RQ_{infer\text{-}pair}$ How effectively can an event inference model infer app problems in response to a user action?

By answering **$RQ_{extract}$**, we determine CASPAR's performance in automatically extracting action-problem pairs, compared to manual annotations. By answering **$RQ_{infer\text{-}pair}$**, we can determine CASPAR's practical value in (1) linking user actions and app problems, as well as (2) inferring possible app problems that may happen after a user action.

1.4.1.2 Background

Post-deployment user feedback, an important facet of user involvement in software engineering, requires developers' close investigation as it contains important information such as feature requests and bug reports [19, 28]. We have found that a user's review of an app often tells a mini story about how the user interacted or attempted to interact with the app. This story describes what function the user tried to bring about and how the app behaved in response. We define a *action-problem* pair as such a pair of events in which an app problem (an event) follows or is caused by a user action (an event). Such event pairs describe where and how the app encounters a problem. Therefore, these pairs can yield specific suggestions to developers as to what scenarios they need to address.

1.4.1.3 Research Plan

We propose a method for extracting action-problem pairs from app reviews. To answer **$RQ_{extract}$**, we investigate the performance of CASPAR in (1) classifying events as USER ACTIONS or APP PROBLEMS, and (2) identifying action-problem pairs compared to a human annotator.

Once event pairs are collected, analyzing them remains a big challenge. We propose the *event follow-up classification* for event inference on action-problem pairs. A classifier determines whether an app problem event is the actual follow-up of a user action or a random event. We develop different models for this classification.

1.4.1.4 Progress

CASPAR discovers high-quality event pairs regarding app problems from reviews, and infers plausible app problems for use cases. We conduct two main evaluations. First, CASPAR classifies the events with an accuracy of 82.0% on manually labeled data. Second, relative to human evaluators, CASPAR extracts event pairs with 92.9% precision and 28.9% recall, and infers events with high plausibility.

By presenting CASPAR, we emphasize the significance of conducting research into user-reported stories in app reviews. These stories, though not without typos and grammatical errors, are valuable deployment logs reported by real users of the applications. Extracting structured stories from user-generated texts and making practical use of them remain challenges that call for deep thinking and more investigations. The resulting paper of this project has been submitted to ICSE 2020. The details and discussions of CASPAR will be presented in Chapter 2.

1.4.2 EPOCI: Ordering Event Pairs based on Commonsense Inference

To address $\mathbf{RQ_{infer\text{-}pair}}$ more generally, we focus on event pairs extracted from everyday stories. We propose the EPOCI task, event pair ordering based on commonsense inference, and methods for the classification of event pairs based on their inferential orders. This task contributes to event inference in that it investigates the inferential relations between each pair of events.

1.4.2.1 Research Plan

$\mathbf{RQ_{infer\text{-}pair}}$: How can we effectively conduct automatic event inference on event pairs to (1) determine their inferential order and (2) infer plausible follow-up events based on one event?

1.4.2.2 Background

Event inference is an important research area of language understanding, and requires understanding the intricate relations between events based on commonsense knowledge. The goal of an event inference task is often to infer plausible follow-up events, given the context events. One important task is to infer an event that has been held-out from the story [4]. The Story Cloze Test [27] is a popular task on held-out event inference, where an inference model infers the last event, i.e., the ending, based on four preceding events, in everyday stories extracted from personal daily weblogs. With recent advances in natural language processing (NLP) and deep learning, event inference is gaining increasing attention. However, this task remains extremely challenging.

Recent studies have started to focus on event inference on shorter sequences of events, namely, event pairs. Event2Mind [36] contributes to event inference by investigating the intents and reactions of a single event. Event2Mind includes a dataset of 25,000 events along commonsense intents and reactions of the participants, collected via crowdsourcing. Their work lacks the transition from intents to actions for event inference. Atomic [40] extends Event2Mind by introducing more dimensions for inferring events with *if-then* reasoning. Their work therefore is also limited to these dimensions for general inference on event pairs.

Some studies define an event as an abstract object that refers to an actual incident that has taken place. Others may treat events as tuples that comprise various attributes that describe events. In our study, we refer to an *event* in a story as a part of a sentence that describes a single action. We use the term *event phrase* to talk about an event as it is represented in language.

Event inference usually involves the investigation of the relations between events. Previous studies focus on two types of event relations, temporal relations and causal relations. Several recent studies have investigated the extraction temporal relations between events based on how the events appear in texts. Mani et al. [24] apply rules and axioms, such as the existence of marker words like *before* and *after*, to infer temporal relations between events. Mirroshandel and Ghassem-Sani [26] extract temporal relations of events from news articles with carefully engineered features, including tense, polarity, and modality, as well as extra event-event features, such as the existence of prepositional phrases.

Other studies have endeavored to extract causal relations based on events' temporal orders. Beamer and Girju [1] propose the concept of *causal potential* as a measure of the tendency of the causal relation between a pair of events. Two events tend to be more causally related if their order appears more frequently than the reverse order. Hu and Walker [15] extract temporal relations of actions from action-rich movie scripts, and infer their narrative causality. Based on similar ideas, Hu et al. [14] extract and infer fine-grained event pairs that are causally related from blogs and film descriptions. Studies of event relation extraction on texts with lower quality, such as tweets and online reviews, are still lacking.

These studies focus on the presentational features of event pairs, rather than the underlying reason why they have such relations. In our research, we investigate the relations between events by understanding the event phrases themselves.

1.4.2.3 Research Plan

To create a dataset of inferentially ordered event pairs, we extract event phrases from short stories, compose them as event pairs, and use human annotators to determine their inferential order. We then propose methods for the classification of event pairs, and report their performance on the EPOCI task.

1.4.2.4 Progress

We have created the EPOCI dataset, a manually labeled dataset of 8,967 event pairs ordered by commonsense inference. Our proposed models, trained on this dataset, have yielded moderate results for the EPOCI task. We have determined that EPOCI is a challenging task.

1.4.3 PAIRS4STORIES: Event Inference in Stories based on Event Pair Inference

To address $\mathbf{RQ_{infer\text{-}story}}$, we plan to study the effectiveness of inferring events in stories based on information learned from inferentially ordered event pairs. The method is tentatively named as PAIRS4STORIES.

1.4.3.1 Research Question

$\mathbf{RQ_{infer\text{-}story}}$: How can we effectively leverage information learned from ordered event pairs to infer events in stories?

1.4.3.2 Background

Leveraging event relations has been proven effective in event inference tasks such as the narrative cloze test [45]. The Story Cloze Test can also be effectively solved by only considering the last two sentences [42]. Certain stories, such as those in the aforementioned app reviews and HHS breach reports, present simple structures, where relations between event pairs are the active propeller for the story progression. In app reviews, action-problem pairs are the major story lines, and in breach reports, breach-remedy pairs are the main theme. However, research on using information of all event pairs in a story for event inference is still lacking.

1.4.3.3 Research Plan

Leveraging information in other event pairs in a story can potentially yield higher accuracy for event inference in stories. Instead of a sequence of events, a story can be considered as a graph of event pairs. Information needed for event inference in stories may reside in the interplay of event pairs. We plan to develop PAIRS4STORIES, a model that learns relations between event pairs and conducts inference in stories.

As success criteria, PAIRS4STORIES should yield higher accuracy for the Story Cloze Test, and practical performance for other event inference tasks.

1.5 Status and Tentative Timeline

Table 1.1 shows the status of our contributions and a tentative timeline for completion.

Table 1.1 Proposed plan.

	Task	Status	Estimate Time
1	CASPAR	Complete	–
2	EPOCI	Almost Complete	Dec 2018–Dec 2019
3	PAIRS4STORIES	Ideation	Dec 2019–Aug 2020

CHAPTER

2

EXTRACTING TARGETED EVENT PAIRS FROM TEXT

We address $\mathbf{RQ_{extract}}$ and $\mathbf{RQ_{infer\text{-}pair}}$ in the setting of action-problem event pairs in app reviews. App users describe their interaction with apps as stories in their reviews. Previous studies on app reviews focus on the classification of the entire reviews. We investigate app reviews on the event level, and focus on the extraction and inference of action-problem pairs, which describe the scenarios where expected user actions trigger unexpected app problems. To this end, we present CASPAR, a method for collecting and analyzing user-reported mini stories regarding app problems from app reviews. CASPAR abstracts event pairs from stories in reviews. By extending and applying natural language processing and deep learning, CASPAR extracts ordered events from app reviews, classifies them as user actions or app problems, and conducts inference on action-problem event pairs. It builds and trains an inference model with the extracted event pairs to predict possible app problems for different use cases. We have evaluated CASPAR, and concluded that it discovers high-quality event pairs regarding app problems from reviews, and infers plausible app problems for use cases. By presenting CASPAR, we demonstrate the importance and effectiveness of extracting targeted event pairs from text.

2.1 Introduction

We motivate the development of CASPAR, and introduce our research questions and contributions.

2.1.1 Motivation

As we have mentioned in Chapter 1, a user's review of an app often tells a mini story about how the user interacted or attempted to interact with the app. This story describes what function the user tried to bring about and how the app behaved in response. These interactions, which we interpret as mini stories, are prominent in reviews with negative ratings. Investigating stories present in app reviews has major implications for software engineering. These stories not only serve as de facto deployment reports for an app, but also express users' expectations regarding the app.

Apps, especially popular ones, receive a large amount of user reviews. Manually combing them is usually impractical. Previous research has studied the classification of app reviews, and automatically identified bug reports, saving collection time for analysts and developers. However, manually reading entire reviews to identify reports of app problems remains time-consuming. Therefore, we motivate CASPAR, a method for collecting and analyzing stories of app problems, as action-problem event pairs, from app reviews. We focus on events in app reviews, and identify the app problem events, which further cut down time needed for the analysts and developers.

A story of interest in this study includes at least two types of events, user actions and app problems. An app *problem* is an undesirable behavior that violates a user's expectations. In particular, when a review gives a negative rating, the stories within it contain rich information regarding app problems. These app problems when reported on (and sometimes ranted about) by users call for a developer's immediate attention. Negative reviews tend to act as discussion points and can be destructive to user attraction and retention.

We define an *action-problem pair* as such a pair of events in which an app problem (an event) follows or is caused by a user action (an event). Such event pairs describe where and how the app encounters a problem. Therefore, these pairs can yield specific suggestions to developers as to what scenarios they need to address.

However, collecting and analyzing action-problem pairs from app reviews is a challenging task. First, extracting the targeted events, i.e., user actions and app problems, is nontrivial—because user-provided texts are heterogeneous and are often riddled with typos and grammatical errors. Second, users may not describe the events of their interaction with apps in a sequential order. The temporal or causal links between events should be investigated. Third, inference on event pairs is known to be a hard problem [36, 48].

2.1.2 Research Questions

As we have mentioned in Chapter 1, CASPAR addresses the modified versions of $\mathbf{RQ_{extract}}$ and $\mathbf{RQ_{infer\text{-}pair}}$, specific to the action-problem pairs in negative app reviews.

$\mathbf{RQ_{extract}}$ How effectively can we extract app problem stories as action-problem pairs from app reviews?

$\mathbf{RQ_{infer\text{-}pair}}$ How effectively can an event inference model infer app problems in response to a user action?

To answer $\mathbf{RQ_{extract}}$, we investigate the performance of CASPAR in (1) classifying events as USER ACTIONS or APP PROBLEMS, and (2) identifying action-problem pairs compared to a human annotator.

Once event pairs are collected, analyzing them remains a big challenge. CASPAR infers app problem events based on user actions. By answering $\mathbf{RQ_{infer\text{-}pair}}$, we can determine CASPAR's practical value in (1) linking user actions and app problems, as well as (2) inferring possible app problems that may happen after a user action.

2.1.3 Contributions

To the best of our knowledge, CASPAR is the first work on app reviews that focuses on the user-app interaction stories that are told in app reviews. CASPAR extracts app problems and the use cases where they happen from user feedback to assist developers in learning about problems in the user experience their apps provide.

In this study, we introduce and provide the first solution to the research problem of identifying and analyzing user-reported stories. CASPAR adopts natural language processing (NLP) as well as deep learning, and brings the investigation of app reviews down to the event level. Instead of generating a selective set of full reviews, CASPAR yields high-quality pairs of user action and app problem events. Moreover, by linking app problem events and user actions, CASPAR can infer probable problems corresponding to a use case. A crucial meta-requirement in app development is to avoid such problems.

Our contributions include: (1) a method for collecting and analyzing stories of app problems, as action-problem event pairs, from app reviews, and (2) a resulting dataset of collected event pairs. By presenting CASPAR, we emphasize the importance of analyzing user-reported stories regarding the usage of applications.

2.2 Related Work

Information residing in user reviews for application is crucial in maintaining and improving software. Analyzing informative reviews and prioritizing feedback have been shown to be positively linked to app success [30]. We now introduce recent work on analyzing app reviews, which mostly involves generic NLP techniques. The task of extracting and analyzing stories in app reviews and applying event inference on those stories have not been addressed. Related work on event inference and event relations has been introduced in Section 1.4.

Previous studies on information extraction from app reviews have emphasized the classification of reviews as a way of combing through the large amount of text and reducing the effort required for analysis.

Pagano and Maalej [29] report on empirical studies of app reviews in the Apple Store. They identify 17 topics in user feedback in app stores by manually investigating the content of selected user reviews. Pagano and Maalej also find that a significant fraction of the reviews, in particular, 96.4% of reviews with 1-star ratings, include the topics of shortcoming or bug report, which should be mined for requirements-related information.

Maalej and Nabil [23] classify app reviews according to whether or not they include bug information, requests for new features, or simply praises. Maalej and Nabil apply classification techniques, such as Naive Bayes and Decision Trees, with features of reviews such as bag of words, ratings, and sentiment, and achieve satisfying results. Based on their classification method, Dhinakaran et al. [8] investigate active learning to reduce manual effort in annotation.

Panichella et al. [31] classify user reviews based on a taxonomy relevant to software maintenance and evolution. The base categories in their taxonomy include *Information Giving*, *Information Seeking*, *Feature Request*, and *Problem Discovery*. The *Problem Discovery* type of app reviews describe app issues or unexpected behaviors. By this classification, they focus on understanding the intentions of the authors of the reviews.

Chen et al. [6] employ unsupervised techniques for identifying and grouping informative reviews. Their framework helps developers by effectively prioritizing and presenting the most informative app reviews.

Guzman et al. [11] investigate user feedback on Twitter, and identify and classify software-related tweets. They leverage Decision Trees and Support Vector Machines (SVMs) to automatically identify relevant tweets that describe bugs, shortcomings, and such.

These studies focus on the analysis of entire reviews and do not dive into the details that the users are describing. With the amount of available app reviews increasing, reading through reviews become more time-consuming. To reduce the time required by developers, recent research targets

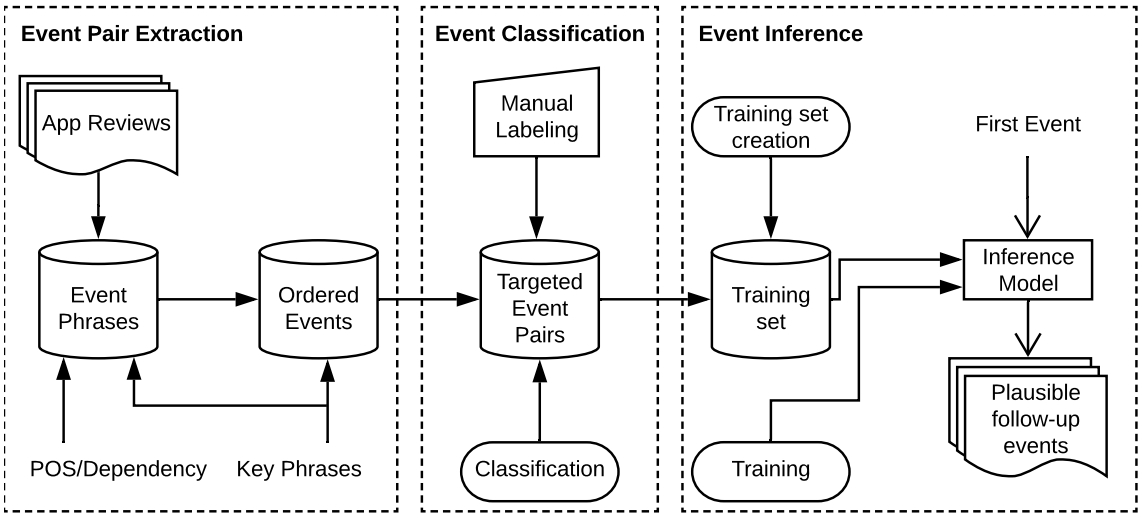

Figure 2.1 An overview of CASPAR.

certain topics, and investigates user reviews on the sentence level. Iacob and Harrison [16] retrieve sentences that contain feature requests from app reviews by applying carefully designed rules, such as keyword search and sentence structures. These rules have been informed by an investigation of the ways users express feature requests through reviews. Di Sorbo et al. [9] summarize app reviews by grouping sentences based on topics and intention categories. Developers can learn feature requests and bug reports more quickly when presented with the summaries. Kurtanović and Maalej [21] classify reviews and sentences based on user rationale. They identify concepts such as issues and justifications in their theory of user rationale. Using classification techniques, Kurtanović and Maalej synthesize and filter rationale-backed reviews for developers or other stakeholders.

2.3 Method

We now present the details of CASPAR, which consists of three steps. First, CASPAR extracts events from targeted app reviews based on heuristics and natural language processing (NLP) techniques. Second, CASPAR classifies the events and keeps the ordered pairs of user action and app problem events. Finally, CASPAR trains an inference model on the extracted event pairs, and infers app problem events given the user actions. Figure 2.1 shows an overview of CASPAR.

2.3.1 Event Extraction

Let us describe the heuristics we adopt to select targeted app reviews, and details regarding the extraction of events.

When an app review reports a problem, it tends to describe the user's interaction with the app as a story, i.e., a sequence of events. However, not all events in a review are necessarily related to each other. In this study, we focus on action-problem event pairs, each of which comprises (1) an expected user action and (2) an app problem that indicates a deviation from expected app functionality. The app problem happens after or is caused by the user action.

To make sure that the extracted events are temporally ordered and casually related, we keep only the reviews that contain temporal conjunctions, including *before*, *after*, and *when*. In addition, we consider key phrases that indicate temporal ordering, such as *as soon as*, *every time*, and *then*. Instead of processing all available app reviews, which are a large dataset, we adopt key phrase search as a heuristic and keep only the reviews that contain at lease one key phrase.

App problems deviate from users' expectations, and therefore are described in reviews with negative ratings. We collect the app reviews with negative ratings, e.g., one-star ratings, and then apply key phrase search to finalize a more refined dataset of reviews.

We then extract ordered events from these reviews using Part-of-speech (POS) and dependency parsing. We refer to an event in the text as a phrase that is rooted in a verb and includes other attributes related to the verb. To identify such event phrase, we can find the subtree rooted in the verb in the dependency-based parse tree from a dependency parser.

Note that a sentence may include multiple verbs, and some of the verbs may belong in the same event. Based on the results of a dependency parser, we consider only verbs that are parsed as ROOT, advcl (adverbial clause modifier), or conj (conjunct). We choose these types of dependency relation tags because they are excellent indicators of events. Since the dependency tree rooted in a ROOT covers all the words in a sentence, we extract the ROOT event phrase from words that are not incorporated in any other event phrases. Punctuation marks at both ends of an event phrase are removed.

Thus, we take the following steps to extract ordered events from the reviews.

1. Find and keep *key sentences*, i.e., sentences that contain the key phrases, and collect the sentences surrounding them (one preceding sentence and one following sentence);
2. Extract event phrases from these sentences;
3. Order event phrases in key sentences using heuristics;

4. Collect other event phrases in the original order in which they appear in the text.

The heuristics we adopt to order the events are shown in Table 2.1, where $e_1 \rightarrow e_2$ indicates that e_1 happens before e_2.

Table 2.1 Heuristics for events in a complex sentence.

Sentence Structure	**Event Order**
e_1, *before / until / then* e_2	$e_1 \rightarrow e_2$
e_1, *after / whenever / every time / as soon as* e_2	$e_2 \rightarrow e_1$
e_1, *when* e_2	$e_1 \rightarrow e_2$, if verb of e_1 is VBG $e_2 \rightarrow e_1$, otherwise

In the case of "e_1, *when* e_2," e_1 happens first most of the time. However, consider the key sentence in Example 3 (for SnapChat[1]).

Example 3

★☆☆☆☆ username2, 09/16/2014

Virus

I love Snapchat. Use it often. But snapchat gave my phone a virus. So I was using snapchat today when all of a sudden my phone screen turned blue and then my phone shut off for 7 HOURS. 7 HOURS. So I had to delete snapchat because it was messing up my iPhone 5c.

We add the heuristic that when e_1 is in continuous tense, i.e., the verb in e_1 is marked as VBG by the POS tagger, e_1 occurs before e_2.

Note that the key phrases are not included within any event phrase. Instead, we label the events based on their positions relative to the key phrases. For example, if an event appears in a subclause led by *when*, it will be labeled as a *subclause* event, and the event outside of this subclause will be labeled as *main*. Events that are not in a key sentence are labeled as *surrounding*. We keep these labels as context information to make the events more readable.

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

2.3.2 Event Classification

In this step, we classify the extracted events into USER ACTIONS, APP PROBLEMS, or NEITHER. To create a training set for the classification, we conducted multiple rounds of manual labeling. The details of the setup are explained in Section 2.4.

We define USER ACTIONS as what the users are supposed to do to correctly use the app, usually an anticipated use case. We define APP PROBLEMS as the incorrect behaviors of an app in response to the user actions (including the lack of a correct response), which do not appear to have been designed by the app developers. In negative reviews, users sometimes complain about the designed app behaviors, which we do not classify as problems. Accordingly, we disregard the types of event phrases shown in Table 2.2, without checking the context (the reviews from which they are extracted). Event phrases that fall into these categories are labeled as NEITHER.

Table 2.2 Types of event phrases we disregard (classify as NEITHER).

	Event phrase type	Example
1.	Event phrases that have been incorrectly parsed	—
2.	Users' affections or personal opinions toward the app	*it has made the app bad, MS OneDrive is superior*
3.	App behaviors that are designed by the developers	*I guess you only get 3 of the 24 levels free*
4.	Users' observations of the developers	*you guys changed the news feed*
5.	Users' requests of features	*needs the ability to enter unlimited destinations*
6.	Users' imperative requests for bug fixes	*fix the app please*
7.	Users' behaviors that are not related to the app	*I give you one-star, I contacted customer service*
8.	Events that are ambiguous without context or too general	*it was optional, I try to use this app*

Before performing the classification, we need to convert the event phrases into a vector representation. One basic encoding method is TF-IDF (term frequency-inverse document frequency) [39], which we adopt as a baseline. However, TF-IDF loses information from the phrase since it ignores the order in which the words appear. To obtain results that are more accurate, we adopt the Universal Sentence Encoder (USE) [3] to convert event phrases into vectors. USE is a transformer-based

sentence embedding model that leverages the encoding subgraph of the transformer architecture [44]. The pretrained USE model and its variants have become popular among researchers for downstream tasks, such as text classification and clustering [47]. The USE vectors capture rich semantic information and can achieve state-of-the-art performance for these tasks.

We then adopt Support Vector Machine (SVM) [38] to classify the sentence vectors into the aforementioned three classes. We adopt two separate classifiers (with probability estimate) for USER ACTIONs and APP PROBLEMs, respectively, since SVM can be applied only on binary classification. If an event is classified as both a USER ACTION and an APP PROBLEM, we choose the class with the higher probability.

Upon obtaining sequences of ordered events, each of which has been classified as a USER ACTION or an APP PROBLEM, we can extract *action-problem pairs* by selecting user actions as well as the app problems that immediately follow them.

We address $\mathbf{RQ_{extract}}$ by reporting the accuracy of the event classification, as well as the precision and recall of the event pair extraction by manually verifying the results of CASPAR applied on a small dataset of app reviews.

2.3.3 Event Inference

The goal of conducting event inference on the extracted event pairs is to infer possible app problems, i.e., unexpected app behaviors, based on an expected user action. Developers can preemptively address possible issues to ensure application quality. We propose treating this event inference task as a classification problem. Given a pair of ordered event, $\langle e_u, e_a \rangle$, where e_u is a USER ACTION and e_a is an APP PROBLEM, the classifier determines whether e_a is a valid *follow-up event* to e_u or a *random event*. Thus, the classes for each entry are ORDERED EVENT PAIR and RANDOM EVENT PAIR. We define this type of classification as *event follow-up classification*.

We need to conduct negative sampling to train a classifier for event follow-up classification. Negative sampling is the process of generating negative observations in scenarios where they are not provided explicitly. The extracted event pairs can be kept directly as positive examples. To create negative ones, we compose an event pair by randomly choosing an app problem for each user action.

In addition to encoding an event into a vector using sentence encoding techniques, we can convert a event phrase into a list of word vectors. Converting words into vectors require a word embedding technique. *Word embedding* is the collective name for models that map words or phrases to vectors of real numbers that represent semantic meanings. Popular word embedding techniques include Word2Vec [25] and GloVe [32].

We experiment with the following classification models.

Baseline: We adopt SVM for this classification problem. As a baseline, we first convert each event into a vector using TF-IDF, and then concatenate the vectors of the two events in an event pair, and train an SVM classifier on the concatenated vectors.

USE+SVM: Instead of adopting TF-IDF, we convert each event into a vector using USE, similarly to the event classification described in the previous section, and then concatenate the USE vectors of the two events in an event pair. We then train an SVM classifier on the concatenated vectors.

Bi-LSTM network: We concatenate the tokens in the two events, separated by a special token, [SEP], convert the concatenated tokens to a sequence of word vectors, and train a bidirectional LSTM network for the classification of the sequences of vectors.

Training set improvement: If the collection of app problem events contains a significant amount of duplicates or similar sentences, a random event from the collection is likely to be similar to the follow-up event, which can impair the accuracy of the classification. We can improve the training set by choosing dissimilar events when composing the negative examples. We consider the following two methods of choosing events for negative examples.

- **Clustering.** We cluster all app problem events into two groups based on cosine similarity, and select the random event for the negative example from the cluster that is different from that of the follow-up event.
- **Similarity threshold.** When choosing the random event for the negative example, we calculate its similarity to the follow-up event. The similarity score should be lower than a threshold. We adopt cosine similarity between the vectors produced for the events, and report the performance of the classifiers using thresholds 0.50 and 0.25, respectively.

The trained classification models can be leveraged for inferring app problems that can follow or be caused by a user action. For a given user action, e_u, we can rank all possible app problems, e_a^i, by the model's confidences of the pair $\langle e_u, e_a^i \rangle$ being an ORDERED EVENT PAIR. The top-ranked app problems can be treated as the results of event inference.

Our observations show that many app problems are similar to each other, for which a classifier should yield similar probabilities. To diversify the inferred events, we choose a similarity threshold, and enforce that the cosine similarity between any two inferred events should be below this threshold.

We address $\mathbf{RQ_{infer\text{-}pair}}$ by reporting the accuracy of the event follow-up classification, and manually verify the plausibility of inferred app problem events.

2.4 Results

To evaluate CASPAR, we need a dataset of app reviews and their ratings. We collected 5,867,198 reviews of 151 apps from 2008-07-10 to 2017-09-15, by crawling the app reviews pages on Apple App Store.[2] As we mentioned in Section 2.3, we focus on reviews with negative ratings. In our experiment, we focused on the 1,220,003 reviews with 1-star ratings. In addition, we applied key phrase search to further filter the dataset. The total number of targeted reviews is 393,755. Table 2.3 lists the key phrases and the count of reviews that contain each of them. Note that some reviews contain multiple key phrases.

Table 2.3 Counts of negative reviews with key phrases.

after	*as soon as*	*before*	*every time*	*then*	*until*	*when*	*whenever*	*while*	Total
77,360	7,603	55,630	53,341	81,338	42,823	152,568	8,563	25,237	393,755

The following experiments are conducted on this refined dataset of negative reviews that contain these key phrases. We will release these reviews, the extracted events, and the action-problem pairs upon publication.

2.4.1 Event Extraction

Following the steps described in Section 2.3, we extract 1,308,188 events from the reviews. We adopt the Python spaCy library[3] for the tokenization, part-of-speech tagging, and dependency parsing of the app reviews. The resulting distribution of event labels is shown in Table 2.4.

The events listed in Table 2.5 are extracted from the review in Example 2. Note that the events have been ordered based on the heuristic regarding *when*.

2.4.2 Event Classification

We conducted three rounds of manual labeling with three annotators that are familiar with text analysis and app reviews. The three annotators were asked to label each extracted event as a USER ACTION, an APP PROBLEM, or NEITHER, as described in Section 2.3. For each round, we randomly

[2] `https://apps.apple.com/us/genre/ios/id36`
[3] `https://spacy.io/`

Table 2.4 Numbers of extracted events with different labels.

Event label	Count
main	396,365
subclause	385,396
surrounding	526,427
Total	1,308,188

Table 2.5 Events extracted from Example 2.

ID	Event label	Event phrase
e_1	*Surrounding*	I 'm going to look for another weather app
e_2	*Subclause*	(*when*) I try to scroll thru cities
e_3	*Main*	It hesitates
e_4	*Surrounding*	I 'm so irritated with this fact alone ...

selected extracted events from the results in the previous step. In the first two rounds, each annotator labeled all events in a subset, followed by the annotators resolving their disagreements through discussion. In the third round, each event was labeled by two annotators, and the disagreements were resolved by labeling the events as NEITHER. We consider this resolution acceptable, as the NEITHER events are not considered in the event inference task.

Manual labeling. Table 2.6 shows the Cohen's kappa for each round of manual labeling between each two annotators before resolutions.

Table 2.6 Pair-wise Cohen's kappa for manual labeling.

Round	Count	Cohen's kappa
1	100	0.630, 0.502, 0.603
2	100	0.607, 0.542, 0.572
3	1,200	0.614

Considering there are three classes (so agreement by chance would occur with a probability of 0.333), the results show that the annotators had moderate to good agreement over the labels before their discussions. After removing some events with parsing errors or those too short (containing

one word), the resulting dataset contains 1,386 labeled events. Table 2.7 shows the distribution of this dataset.

Table 2.7 Distribution of the manually labeled dataset.

Event type	Count
USER ACTION	401
APP PROBLEM	383
NEITHER	602
Total	1,386

Classification. We adopt the Universal Sentence Encoder in TensorFlow Hub[4] to encode each event phrase into a vector. Each USE vector is of size 512. As described in Section 2.3, we train two SVM classifiers for a three-class classification. We adopted the SVM implementation of scikit-learn,[5] which provides an estimate for the probability of a classification. One SVM classifies an event into USER ACTION or other; the other SVM classifies an event into APP PROBLEM or other. For a given event, e, the first SVM yields a probability, u, of e being a USER ACTION, and the second SVM yields a probability, a, of e being an APP PROBLEM. We adopt the following formulae to convert these probability estimates into a three-class probability distribution. Each tuple below is of the form, P_{NEITHER}, P_{ACTION}, and P_{PROBLEM}, which represent the probability estimates of event e being NEITHER, a USER ACTION, or an APP PROBLEM respectively.

If $u \geq 0.5$ and $a \geq 0.5$,

$$P(e) = \left(\frac{2(1-u)(1-a)}{2-u-a+2ua}, \frac{u}{2-u-a+2ua}, \frac{a}{2-u-a+2ua}\right)$$

If $u \geq 0.5$ and $a < 0.5$,

$$P(e) = \left(\frac{1-u}{1+a}, \frac{u}{1+a}, \frac{a}{1+a}\right)$$

If $u < 0.5$ and $a \geq 0.5$,

$$P(e) = \left(\frac{1-a}{1+u}, \frac{u}{1+u}, \frac{a}{1+u}\right)$$

[4] https://tfhub.dev/google/universal-sentence-encoder-large/3
[5] https://scikit-learn.org/stable/

If $u < 0.5$ and $a < 0.5$,

$$P(e) = (\frac{0.5}{0.5+u+a}, \frac{u}{0.5+u+a}, \frac{a}{0.5+u+a})$$

The purpose of this exercise is to convert two probability estimates into a three-class probability distribution via continuous transformation while preserving the results of the original classifiers. An event is classified into the class with the highest probability after this transformation.

We use 90% of the dataset for training and 10% for testing. We report the performance arising from 10-fold cross validation for each classifier. The results are shown in Table 2.8.

Table 2.8 Accuracy of event classification.

Classification	TF-IDF	USE
USER ACTIONS vs. Others	81.2%	86.9%
APP PROBLEMS vs. Others	80.2%	86.4%
USER ACTIONS vs. APP PROBLEMS vs. NEITHER	71.2%	82.0%

We then apply the trained classifiers to the entire dataset of extracted events. Table 2.9 shows the results for the events extracted from Example 2.

Table 2.9 Event classification for events in Example 2.

ID	Event phrase	$P_1(e)$	$P_2(e)$	Prediction
e_1	I 'm going to look for another weather app	0.212	0.057	NEITHER
e_2	I try to scroll thru cities	**0.939**	0.022	USER ACTION
e_3	It hesitates	0.034	**0.705**	APP PROBLEM
e_4	I 'm so irritated with this fact alone …	0.036	0.080	NEITHER

Event pairs. All adjacent and subsequently ordered action-problem event pairs are then collected for event inference. For example, $\langle e_2, e_3 \rangle$ in Table 2.9 is collected accordingly. The total number of extracted event pairs is 85,099. Some other example (some paraphrasing to save space) for the same app can be found in Table 2.10.

Table 2.10 Extracted event pairs for the Weather Channel.

User Action		App problem
(after) I upgraded to iPhone 6	→	this app doesn't work
(as soon as) I open app	→	takes me automatically to an ad
You need to uninstall app	→	(before) location services stops
(every time) I try to pull up weather	→	I get "no data"
(whenever) I press play	→	it always is blotchy
(when) I have full bars	→	Always shows up not available
I updated my app	→	(then) it deleted itself

To evaluate the effectiveness of CASPAR in extracting action-problem pairs, we selected a random sample of 200 app reviews of rating 1, and asked a human annotator to manually extract such event pairs. Of these 200 reviews, only 63 of them contain at least one key phrase that we have adopted. Based on whether an event pair has been identified, we compose two confusion matrices of this study, one for all reviews and one for reviews with key phrases only, as shown in Table 2.11.

Table 2.11 Manual verification of CASPAR extraction results.

		All reviews		Reviews w/ key	
		Human		Human	
		ID-ed	Not ID-ed	ID-ed	Not ID-ed
CASPAR	ID-ed	13/200	1/200	13/63	1/63
	Not ID-ed	32/200	154/200	19/63	30/63

If we consider the human results as the ground truth, CASPAR has an overall accuracy of 83.5%, a precision of 92.9%, and recall of 28.9%. Of the 45 reviews in which the human annotator has identified event pairs, 32 reviews (71.1%) contain at least one key phrases. We discuss these results in Section 2.5.

2.4.3 Event Inference

We divide the extracted pairs into a training set (90%) and a testing set (10%). We report the accuracy of each method for the event follow-up classification.

To convert each token into vectors, We adopted one of spaCy's pre-trained statistical models for

English, en_core_web_lg[6], with GloVe vectors [32] trained on Common Crawl[7] data. Each GloVe vector is of size 300. We implemented the bidirectional LSTM network using TensorFlow.[8] The size of the hidden layers is 256. An Adam Optimizer [17] with learning rate of 0.0001 is used to minimize the sigmoid cross entropy between the output and the target. We trained the model for 20 epochs with batch size of 1.

To improve the quality of the training set, we adopted two types of rules during negative sampling, namely, clustering and similarity threshold. For clustering, we adopted KMeans in scikit-learn to divide all app problem events into two clusters ($k = 2$) based on their cosine similarity. For the similarity threshold method, when choosing a random event for as a negative example, the cosine similarity between that event and the follow-up event has to be below the threshold. We experiment with thresholds of 0.5 and 0.25, respectively, and report the performance of the classification. All cosine similarity calculations have been conducted on the USE vectors of the events.

The results are shown in Table 2.12. The training set column identifies the method for constructing the negative examples.

Table 2.12 Accuracy of classification of event pairs.

Classifier	Training set	Accuracy
Baseline	Random	55.3%
USE+SVM	Random	66.0%
Bidirectional-LSTM	Random	67.2%
Baseline	Clustering	58.5%
USE+SVM	Clustering	67.8%
Bidirectional-LSTM	Clustering	67.8%
Baseline	Similarity < 0.5	60.7%
USE+SVM	Similarity < 0.5	68.1%
Bidirectional-LSTM	Similarity < 0.5	69.1%
Baseline	Similarity < 0.25	72.9%
USE+SVM	Similarity < 0.25	82.8%
Bidirectional-LSTM	Similarity < 0.25	79.6%

For event inference, the input should be a USER ACTION. We rank all possible APP PROBLEM

[6] https://spacy.io/models/en
[7] http://commoncrawl.org/
[8] https://www.tensorflow.org/

events by their probabilities of being a valid follow-up of the input, and consider the top-ranked ones as the results of the event inference. We choose a similarity threshold of 0.75 to diversify the inferred events. We consider only app problems extracted from reviews of the same app as the input.

Figure 2.2 shows the top-10 app problem events for the user action *I try to scroll thru cities* by the trained bidirectional LSTM network (trained with improved training set with Similarity < 0.25). The inferred event *it loads for what seems like forever* presents the most similar meaning to the ground truth.

We ask three human annotators to independently label each of these output app problems events based on whether it is plausible that it follows or is caused by the user action. All three annotators labeled a_1, a_2, a_3, a_4, and a_8 as plausible (50%), and a_5, a_7, and a_{10} as implausible (30%). They disagreed over the other two events.

2.5 Discussion

We presented CASPAR, a method for collecting and analyzing app problem stories in user-generated app reviews. We now discuss its merits and limitations.

2.5.1 Merits

Previous studies have been focused on text analysis of app reviews on the review level, the results of which are collections of reviews that require developers' further investigation. CASPAR dives deeper to the event level, and can extract and infer app problems, which can help developers improve their applications by addressing and avoiding the problems.

App problem event pairs. By extracting and collecting action-problem pairs from app reviews, CASPAR presents application problems that require attention to the developers in a readable way. The extracted event pairs describe the apps' unexpected behaviors as well as the context, i.e., the users' actions that have triggered them. CASPAR can also be applied to selective app reviews, such as those for certain apps in a period of time, so that the extracted event pairs are more appealing to a particular audience of developers.

By answering $\mathbf{RQ_{extract}}$, we have shown that CASPAR extracts targeted event pairs effectively, and the classification of event types yields high accuracy.

Event inference. By conducting inference on the extracted event pairs, we endeavor to establish a connection between the user actions and app problems. Inferring possible app problems based on a user action can help developers preemptively avoid app issues.

USER ACTION: I try to scroll thru cities
Ground truth: it hesitates
Inferred APP PROBLEMS:

	Unanimously judged plausible
a_1	it says there is an error
a_2	it loads for what seems like forever
a_3	it tells me the info for my area is not available
a_4	the app crashes
a_8	it reset my home location
	Conflicting judgments
a_6	it rarely retrieves the latest weather without me having to refresh
a_9	it goes to a login screen that does not work
	Unanimously judged implausible
a_5	the radar never moves , it just disappears
a_7	I rely heavily on it & for the past month , it says temporarily unavailable
a_{10}	Radar map is buggy – weather activity stalls , appears , then disappears

Figure 2.2 Inferred app problem events to follow-up a user action (threshold = 0.75).

By addressing $\mathbf{RQ_{infer\text{-}pair}}$, we have shown that CASPAR yields satisfactory performance when determining whether an app problem is random or a valid follow-up of a user action. In addition, CASPAR generates plausible follow-up app problems to user actions.

2.5.2 Limitations

CASPAR is not without its limitations.

Key phrases. We target only the app reviews that contain the selected key phrases, which indicate the temporal ordering of events. Using key phrase search has limited the size of the resulting dataset. We use these key phrases because we need the extracted events to be temporally and causally related. Further investigation on how to extract related events is still required, which we leave to future work. In fact, during the manual verification of the performance of event pair extraction, the human annotator identified event pairs that are linked by key phrases like *ever since*, *if*, and *any time* that were missing from our list. Experimenting with key phrases that indicate conditional or causal relations, such as *if* and *because*, is a promising direction. Additional key phrases may be found in a semi-supervised fashion.

Text quality. CASPAR extracts events using a part-of-speech tagger and a dependency parser, which are not reliable when the texts have typos, missing punctuation, or grammatical errors. During the manual verification, human annotators have identified event pairs that CASPAR is not able to parse. For example, one review says *App is now crashing* everyone *I tap a story*, where the typo causes CASPAR to miss the event pairs. Consider the review in Example 4 (for The Weather Chanel).

Example 4

★☆☆☆☆ username3, 05/01/2014

Terrible

Worse than ever. They took an amazing weather app & turned it into crap! U add a location it's automatically added to favorites. Very annoying & time consuming when u are traveling to delete locations. Very busy presentation & not as user friendly. Not a big fan, thanks for NOTHING on the new update!

CASPAR identifies *U add a location it's automatically added to favorites* as one single event, since it is grammatically incorrect, which is classified as NEITHER. However, the human annotator can easily identify this sentence as a pair of events. We estimate the quality of user-generated texts, or the lack thereof, is the most important reason for the low recall of CASPAR in extracting event pairs. In future work includes extraction methods that rely less on the correctness of the parser employed.

Manual labeling. Regarding manual event classification, the annotators achieved only moderate

to good agreement, and the agreement does not improve with additional iterations and discussions. We have excluded data points on which the annotators disagreed by labeling them as NEITHER, which limited the number data points available for event inference.

We identify the following reasons for the disagreement among annotators. First, user-generated texts are prone to typos and grammatical errors, which cause a parser to produce erroneous events. For example, one very common challenge is the omission of proper punctuation. The key sentence in Example 4 is extracted as one event which can be both user action and app problem. Second, events have been stripped out of context, and some lose critical information. For example, the event *reset my phone* is usually a user action, but the annotators could not be sure without context. Example 5 shows the entire review (for Messenger[9]).

Example 5

★☆☆☆☆ username4, 01/22/2016

Great but.....

This is a great app. But it has been crashing before it can load. Reset my phone, got the new update for iOS and it just keeps crashing. Not sure if I'm the only one with this problem.

Third, there are always unforeseen cases where annotators may disagree. For example, the event *switching between apps doesn't make anything faster* can be interpreted as an app problem or an irrelevant event.

Action-problem pairs. CASPAR targets only those event pairs that describe single iterations of user-app interaction. However, this type of interaction does not cover all scenarios of app problems. Some app problems may occur without users' actions. In fact, most app reviews that describe bug reports do not describe in detail the users' actions that caused the unexpected behaviors. We did not focus on such reviews, since they do not provide insightful information for developers to address the problems.

Meanwhile, user-app interaction may include a longer sequence of events than just a pair. For example, the review in Example 5 describes multiple user actions, none of which seemed to have caused the observed problem. However, this review does report a bug that requires developers' attention. The information contained in this review is also potentially useful to the developers.

Many app reviews additionally describe user expectations, user reactions to app problems, or misuses of the apps. We leave the extraction of other formats of user-app interaction to future work.

Event inference. The proposed classification of event pairs yields moderate results. One major reason is that there are quite a few duplicate app problems. For example, the events *app crashed*

[9] `https://apps.apple.com/us/app/messenger/id454638411`

and *app freezes* are common occurrences. A random app problem, which counts as a negative instance, is likely to be semantically similar to the actual app problem. We have proposed methods to improve the training set, which has improved the performance of the classifiers evaluated.

Homogeneity among the extracted events interferes with inference of app problems. The top-ranked events can be similar to each other. To mitigate this problem, we have excluded app problems that are similar to event phrases that are already inferred.

A second possible reason for the moderate performance is that the training set is fairly small, especially for a deep learning model. As we mentioned above, CASPAR adopts fairly strict key phrase search to ensure the quality of extracted pairs. We have collected 85,099 event pairs for 151 different apps, which may not be large enough for the bidirectional LSTM network. We plan to apply CASPAR on app reviews from other application distribution platforms to extract and collect more event pairs. We will also investigate other reliable techniques for the extraction, as future work.

Third, we have simplified event inference to an event follow-up classification, which limits the inference to app problem events that have been reported. To fully infer follow-up events of user actions, we may need to build more sophisticated inference models, such as sequence to sequence models [43]. We leave the investigation of such models to future work.

2.5.3 Threats to Validity

The first threat is that our annotators may lack the expertise in the software development of iOS applications. Our annotators are familiar with or experts on concepts of NLP and machine learning, but they may not possess enough experience in industry, which may have affected their disagreement over the labels.

Second, all of our labeling and training have been conducted on reviews with 1-star ratings from Apple's App Store. Our work may not be generalizable to other reviews where the descriptions of apps' behaviors are not limited to app problems.

Third, in the event inference step, we improve the training set by imposing certain rules regarding the random event. However, we adopt USE vectors for the clustering and calculating similarity, which may have affected the performance of the classification models that leverage the same vectors. We can mitigate this threat by using a different sentence embedding technique for the creation of the training set.

2.6 Conclusions and Future Work

We presented CASPAR, a method for extracting, collecting, and analyzing app problem stories, as action-problem event pairs. CASPAR adopts heuristics and classification and effectively extracts ordered event pairs. By extracting and collecting such app problem instances, CASPAR helps developers by presenting readable reports of app issues, which require their attention. CASPAR extracts high-quality action-problem pairs with a high precision. In addition, CASPAR trains an inference model with the extracted event pairs, leveraging NLP techniques and deep learning models, and infers possible follow-up app problems based on user actions. Such inference allows the developers to preemptively address possible app issues, which helps ensure the quality of the applications.

By presenting CASPAR, we emphasize the significance of conducting research into user-reported stories in app reviews. These stories, though not without typos and grammatical errors, are valuable deployment logs reported by real users of the applications. Extracting structured stories from user-generated texts and making practical use of them remain challenges that call for deep thinking and more investigations.

Future work includes applying more heuristics for event pair extraction, exploring extraction methods that rely less on parsers, studying other types of user-app interaction, and investigating other suitable models for the event inference task.

CHAPTER

3

COMMONSENSE INFERENCE ON EVENT PAIRS

In Chapter 2, we have addressed $\mathbf{RQ_{infer\text{-}pair}}$ in the domain of action-problem pairs in negative app reviews. To address $\mathbf{RQ_{infer\text{-}pair}}$ in a more general sense, we propose the EPOCI task, event pair ordering based on commonsense inference. We define the *inferential order* between events, as the natural sequential order of two events in the same context that can be inferred by commonsense. We have extracted event pairs from everyday stories and identified their inferential orders through manual annotations. Based on the resulting dataset, the EPOCI dataset, we propose and train multiple models for the ordering of event pairs, and compare their performance. Our proposed network, Storyline, yields the highest accuracy for this task in our experiments. The EPOCI task, though remains to be challenging one, contributes to inference on event pairs by investigating the plausibility of one event following the other.

3.1 Introduction

The occurrence of an event often presupposes certain facts and yields plausible implications. For example, *eating an apple* assumes prior possession of an apple, while *buying an apple* implies

subsequently possessing an apple. A story, in the sense of natural language processing, is a sequence of events. Understanding events is the basic requirement of understanding a story. A human with commonsense can understand the presuppositions and implications of an event, which are not usually explicitly expressed in stories. A direct result of having such commonsense is the ability to immediately judge the natural order of two events, assuming these two events are from the same context of a coherent story.

We propose EPOCI task, event pairs ordering by commonsense inference, which is a starting point for understanding the underlying information of events. We consider the *inferential relation* between two events. Two events, e_1 and e_2 are *inferentially orderable* if their sequential order can be inferred based on commonsense alone, assuming they are from the same coherent story. We write

$$e_1 \rightarrow e_2$$

if e_1 can be inferred to have happened before e_2, or e_1 *precedes* e_2 and e_2 *follows* e_1. For example, *buying an apple* $\rightarrow$ *eating an apple*.

Our work is different from previous studies on *temporal relations* between events in that we focus on the innate orders of singleton events instead of the statistical orders that manifest in texts such as books and movie scripts.

As units of analysis, we consider *event phrases*, which are textual phrases, namely sentences or sub-clauses, that contain basic information of an event. We extract event phrases from short stories, compose them as event pairs, and use human annotators to determine their inferential order. In addition, we propose methods, as well as a new neural network model, for the classification of these event pairs, and report their accuracy in determining the inferential orders between them. Our proposed network, Storyline, yields the highest accuracy for this task in our experiments. We conclude that this classification is also challenging.

The EPOCI task contributes to event inference in that it investigates the plausibility of an event following another. With enough training data, we can leverage the results of this task to infer the plausible follow-up events given a story. Our contributions include (i) the EPOCI dataset, a manually labeled dataset of event pairs ordered by commonsense inference; and (ii) evaluation of the proposed methods for the EPOCI task.

3.2 Related Work

As we have introduced in previous chapters, some studies on event relations focus on the external statistics of how event pairs appear in texts, such as news articles and movie scripts. Recent work

has started to investigate the internal meanings of the events. Kober et al. [20] study what event phrases entail based on their tenses and aspectual auxiliaries. Other studies investigate what can be reasoned from one event in different inferential dimensions [36, 40].

We consider events as textual phrases or sentences that describe an incident, as opposed to annotated objects or tuples abstracted from texts [4, 33, 41]. Many existing studies treat events as raw sentences or phrases, and conducts inference directly on natural language. Pichotta and Mooney [34] apply their LSTM-based statistical scripts model on the sentence level, and infer held-out sentences. They train their model on English Wikipedia texts and only achieve moderate results. Other research has focused on better-presented textual stories. Wanzare et al. [46] collect script-specific event sequences for different scenarios describing typical everyday activities through crowdsourcing. The collected script knowledge is helpful to event inference on scripts. The ROCStories dataset published along with the Story Cloze Test Mostafazadeh et al. [27] contains thousands of everyday stories, each of which is composed of five short sentences. The progressions of these stories are mostly unsurprising, and follow commonsense thinking. We adopt this dataset in this study. Srinivasan et al. [42] adopt a simple feed-forward network and skip-thought embeddings to determine whether a sentence is the correct or a random ending to the evidence event sequence, and achieve high accuracy. Interestingly, they have found that, considering only the last sentence instead of all four, i.e., sentence-to-sentence inference, yields higher accuracy.

Many studies focus on pairs of sentences instead of a single piece of text. The Stanford Natural Language Inference (SNLI) [2] dataset classifies pairs of sentences as entailment, contradiction, and neutral. Event ordering differs from entailment because it does not investigate the necessity of the events. We assume that the two events occur in the same coherent story, and try to determine their order based on commonsense inference. SWAG [48] is a large dataset of multiple choice questions, in which the correct choice corresponds to the following event of the event in the question. However, events in SWAG are automatically collected long sentences with rich information that may influence the inference. In our work, we adopt on simpler atomic events and manual annotations, such that a model for the EPOCI task will focus on inferential relations between events.

3.3 Method

We now introduce our methodology in creating the dataset of event pairs, and the proposed methods for the EPOCI task.

3.3.1 Obtaining the Dataset

We need a dataset consisting of event pair candidates, pairs of event phrases that are potentially orderable based on commonsense inference, along with whether they are actually ordered. We build our dataset as follows:

- Extract events from sentences in short stories in ROCStories, and convert them to event phrases;
- Create event pairs based on their original positions in stories, and randomize their orders;
- Use human annotators to label them based on their inferential relations; and then
- Verify the results and compile the dataset.

We now describe each step in detail.

3.3.1.1 Event Extraction

We take the following steps to extract event phrases from ROCStories.

1. From the dependency-based parse tree produced by a dependency parser, extract event phrases, as parts of a sentence that are in the subtrees rooted in verbs that are:
   (a) parsed as ROOT, advcl (adverbial clause modifier), or conj (conjunct); and
   (b) not non-event verbs, such as *want*, *love*, and *need*.
2. Remove indicator words for presuppositions, such as *again*, *also*, and *still*.
3. Clean the subject by:
   (a) adding the subject in the root sentence to each event phrase that does not include a subject; and
   (b) changing person names in the subject into pronouns, such as *he*, *she*, and *they*, based on the results from an NER (named entity recognition) tagger.
4. Clean the verb by removing its tense (they are marked by parentheses).

3.3.1.2 Event Pair Candidates

After the above steps, each story has been converted to a list of event phrases. To create event pair candidates, we select pairs of event phrases that are next to each other, or not too far apart (at most two events between them). We limit the number of event pair candidates in a story to five.

3.3.1.3 Manual Annotations

We then use human annotators to label (a subset of) the event pair candidates. Each annotator is shown two event phrases at a time, and then asked "does it make more sense naturally that e_1 happens before e_2 than that e_2 happens before e_1?" An annotator is tasked to label them with a number, 1 for yes (first event happening before second one), 2 if second event should happen before the first one, and 0 for unable to decide or both orders are equally acceptable. Additionally, annotators are asked to mark the event phrases that are incorrectly extracted, which are later excluded from the final dataset.

We instruct the annotators to label the data points based on their first instinct. The event phrase pairs shown to the annotators are usually out of context or lacking certain information. Orders of events can be different based on different assumptions of the missing information. We instruct the annotators to label the event pair as 0 when they are not sure.

3.3.2 Classification

Determining the inferential order of two events can be considered as a three-class classification problem. We evaluate the following classification methods and report their accuracy.

3.3.2.1 SVM

Similar to the inference model in CASPAR, we convert each event phrase into a vector, concatenate the two vectors in an event pair, and classify it using a Support Vector Machine (SVM) classifier. To encode the event phrase, we adopt average word vectors and the Universal Sentence Encoder (USE) [3].

For word embedding, we apply both GloVe Pennington et al. [32] and Google's pre-trained Word2Vec model Mikolov et al. [25].[1]

3.3.2.2 The Storyline Network

We propose a simple neutral network, *Storyline*, for the general purpose of inference on event pairs. We consider a vector space for events where the progression from one event to the next one can be represented by linear transformation.

One of the major obstacles of event inference is that one event, or the *core event*, may have numerous plausible follow-up events that are semantically different from each other. Previous studies investigate *inferential dimensions* with well-defined meanings as directions for inference.

[1] https://code.google.com/archive/p/word2vec/

Example 6 shows the events that can be inferred in various inferential dimensions. The events are selected from the EPOCI dataset and the ATOMIC dataset[40] with edits for uniformity.

Example 6

Core event: PersonX (make) fun of PersonY.

Events from EPOCI	Events from ATOMIC
Class 1: PersonY (lose) confidence.	**oEffect**: PersonY (break) up with PersonX.
Class 1: PersonY (push) PersonX down.	**oReact**: PersonY (feel) ashamed.
Class 2: PersonY (go) fishing with PersonX.	**oWant**: PersonY (want) to run away.
Class 2: PersonY (wear) thick glasses.	**xIntent**: PersonX (intend) to annoy PersonY.
Class 0: PersonY (decide) to try PersonY's best.	**xReaction**: PersonX (feel) superior.
Class 0: PersonY (take) many lessons.	**xNeed**: PersonX (need) to find something embarrassing about PersonY.

To determine the inferential order of an event pair, we consider the first event as the *core event*, and the second as the *peripheral event*. The core event contains the information for plausible follow-up events, following their respective story lines. Instead of pre-defined inferential dimensions, we consider implicate story lines, along each of which the core event is plausible to progress. The peripheral event contains the information needed for choosing a compatible story line, such as feelings and reactions. If the peripheral event is a plausible follow-up of the core event, its should align well with the *forward* progression along a compatible story line.

Let $e_c \in R^D$ and $e_p \in R^D$ be the vector representations of the core event and peripheral event, respectively, in the story vector space, where D is the dimension of the vectors. Let N be the number of possible story lines. The follow-up events for e_c can be represented by $\hat{F} = (\hat{f_1}^T, \hat{f_2}^T, \ldots, \hat{f_N}^T)^T \in R^{N \times D}$. We check the compatibility of e_p with each story line by calculating $w_p \in R^N$, a weight vector that represents the weights of e_p over the N story lines. The follow-up event of e_c along the compatible story line of e_p can be represented by:

$$\hat{f}_{c,p} = (x_i | x_i = \Sigma_{j=1}^{N} (w_p[j] \times \hat{F}_{ji})), i = 1, \ldots, D$$

If e_p is the a plausible follow-up event of e_c, e_p should align well with $\hat{f}_{c,p}$. The alignment can be quantified by $e_p \cdot \hat{f}_{c,p}$.

The process of *story line generation* calculates $\hat{F}$ from e_c and *story line picking* generates w_p from e_p. Figure 3.1 shows the overview of the forward compatibility estimation process.

In this work, we assume linear transformations for these processes. For story line generation,

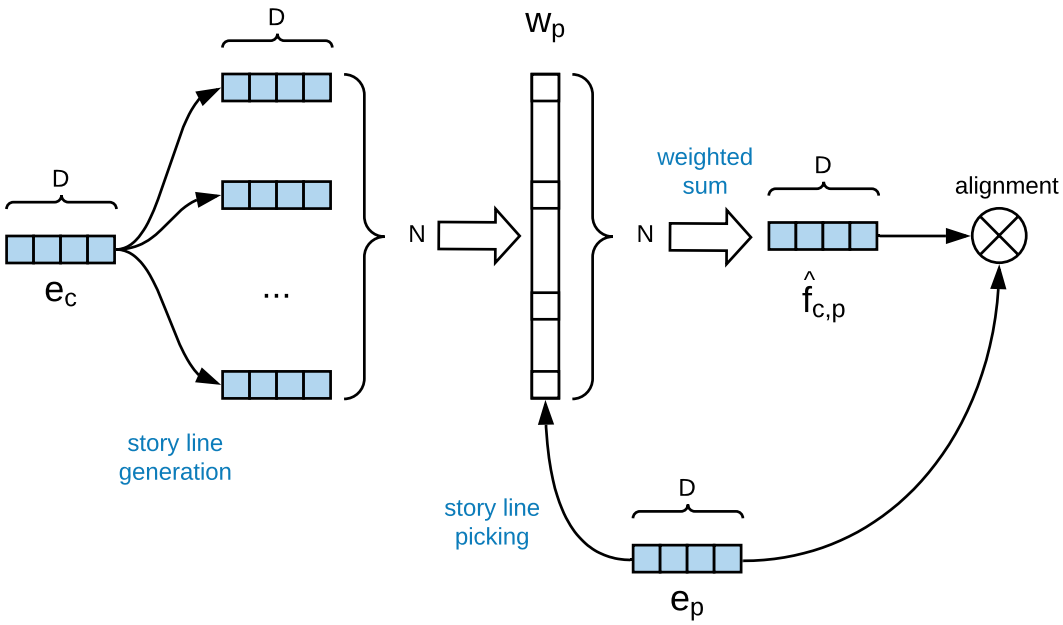

Figure 3.1 Overview of the process of estimating the forward compatibility between events.

we have,

$$\hat{f}_i = \hat{F}[i] = e_c \times W_f[i] + B_f[i], i = 1, \ldots, N$$

W_f is a set of N matrices in $R^{D\times D}$, and B_f is a set of N biases in R^D. The linear transformation with $W_f[i]$ and $B_f[i]$ transforms e_c to $\hat{e}_f^i$, the predicted follow-up event along the i-th story line.

For the story line picking process, we have,

$$w_p = \text{softmax}(e_p \times W_p + B_p)$$

The weights W_p is in $R^{D\times N}$ and B_p is in R^N. We add a softmax function to normalize this vector and emphasize the important story lines.

Backward progression: Symmetrically, we can represent the backward progression of the stories the same way, and we can calculate the alignment of e_p with the predicted backward event $\hat{b}_{c,p}$ as $e_p \cdot \hat{b}_{c,p}$.

Intuitively, if event x happens before event y in the natural order, then $e_y \cdot \hat{f}_{x,y}$ should be significantly greater than $e_y \cdot \hat{b}_{x,y}$, and vice versa. However, if either of them can happen first, both

numbers should be high. If these events are not related, both numbers should be low.

To encode the events in the event vector space, we can use linear transformation of other types of sentence encoding techniques. We report the performance of the Storyline network with average word vectors, Skip-thought vectors[18], and USE.

3.3.2.3 Sequence Classification

A pair of event phrases, when concatenated, yields a sequence of tokens. We consider the following sequence classification methods.

BERT with Fine-Tuning Bidirectional Encoder Representations from Transformers, or BERT Devlin et al. [7], is a language representation model that has performed well on numerous NLP tasks. BERT improves fine-tuning based approaches Radford et al. [35] by using a bidirectional Transformer and "masked language model" as the pre-training objective. The pre-trained BERT model can be fine-tuned with a simple output layer to work on a variety of tasks, such as question answering and language inference. In our experiments, we fine-tuned the pre-trained BERT model for sequence classification.

Bidirectional LSTM Network: Long Short-term Memory (LSTM) networks Hochreiter and Schmidhuber [13] are widely used for NLP tasks. Bidirectional LSTMs have been proven greatly effective on sequence classification. We adopt a straightforward bidirectional LSTM network for our event ordering task.

Note that we use a separation token to denote the sentence boundaries. The output is a softmax layer on the final hidden state (i.e., output of the network) with regard to the first token, which is a three-dimensional vector, corresponding to three classes, namely, undetermined, $e_1 \rightarrow e_2$, and $e_2 \rightarrow e_1$. The model is trained to minimize the cross entropy between the output and the class labels.

3.4 Results

We report the statistics of the final annotated dataset of event pairs, as well as the performance of each baseline over the event ordering task.

3.4.1 The Dataset

We randomly selected 9,000 event pair candidates for manual annotation. Thirty-three candidates were discarded due to parse errors or ambiguity, based on feedback from the annotators. The dataset (available at `https://shorturl.at/ghs04`) of ordered event pairs currently includes

8,967 distinct pairs of event phrases and their annotations. These event phrase pairs were generated based on the ROCStories corpus. Table 3.1 shows the statistics of our dataset.

Table 3.1 EPOCI dataset statistics.

Class	Count
undetermined (0)	1,963
$e_1 \rightarrow e_2$ (1)	3,588
$e_1 \rightarrow e_2$ (2)	3,416
Total	8,967

3.4.1.1 Human Annotations

Forty-nine computer science graduate students provided the annotations. Each event pair has been labeled by three distinct annotators. The final label is determined by majority voting. If there is no majority label, the event pair will be labeled as Class 0.

We calculate the Fleiss' kappa to measure the inter-rater disagreement among the annotators. The annotators achieved moderate agreement (kappa=0.497).

3.4.2 Classification Performance

We have divided the EPOCI dataset into a training set (90%) and a testing set (10%). When training the models, we duplicate the event pairs in the training set based on the notion that, if a pair (e_1, e_2) is in the dataset with a label of 1, (e_2, e_1) can be added to the dataset with a label of 2.

Each model was trained for 20 epochs via Adam Optimizer with a learning rate of 5e-4 and batch size of 10. We used the default settings for BERT. We implemented the bidirectional LSTM (biLSTM) networks using TensorFlow. [2] The number of the hidden states was set to 256. For the Storyline network, the number of dimensions is 16, the dimension for the event vectors is 128.

Table 3.2 shows the accuracy of the three-way sequence classification task of each method described in Section 3.3.2.

[2] https://www.tensorflow.org/

Table 3.2 Accuracy of each classification method.

Classifier	**Accuracy**
Random guessing	35.5%
SVM + GloVe	51.4%
SVM + Word2Vec	52.9%
SVM + USE	52.3%
Storyline + GloVe	47.1%
Storyline + Word2Vec	52.6%
Storyline + Skip	51.7%
Storyline + USE	**54.1%**
BERT	42.3%
biLSTM + GloVe	53.7%
biLSTM + Word2Vec	53.1%

From these results we can tell that the Storyline network with USE yields the highest accuracy. However, its performance is still far from practically viable.

3.5 Discussion

We have motivated the event ordering task, described the creation of the dataset, and demonstrated the results of possible baseline methods for the task. We now discuss the merits and weaknesses of our study.

3.5.1 Event Phrases

We consider only simple event phrases. The motivation comes from the need for understanding coherent flows of simple events in everyday stories, based on commonsense, such as *training for a race* → *entering a race* → *winning a race*, which is the basic requirement of event inference tasks. Simple event phrases are easier to model, and more generalizable in subsequent use.

However, events in ordinary text are usually much richer than event phrases. In addition to the verb phrases, information residing in subjects, objects, and other clauses influences the meaning of

the event. Understanding and analyzing such complicated events therefore requires much more information from the world, which places too much burden on a language model. For example, consider this sentence from a news article: *French-trained Dunaden wins prestigious Melbourne Cup race on Tuesday*. This sentence is complex, but has the same flow of events mentioned above.

3.5.2 The EPOCI Task and Human Annotations

We have proposed the event ordering task, contributing to the understanding of events, including their presuppositions and implications. Our assumption is that humans can tell the inferential order between two related events. Even though it is the case ideally, human annotations are not as reliable as one may hope. Not all related events are immediately orderable, which should be considered for future annotation effort.

Relations between events may be richer than simple *before* and *after*. For example, events may have different durations. One event may be a subevent of another. Actual story-telling can also result in events with surprising orders, and the most natural way to order events may not be inferential ordering. Stories usually follow the order of the protagonist's perceptions of the events, instead of the events actual order.

Our annotators were asked to label the event pairs based on their first impression. However, the first impressions might not be the most probable orders. For example, an annotator may determine an order upon first impression, when both orders are plausible. In future work, annotators should be asked to consider both orders before deciding on an order.

3.5.3 The Dataset

The obvious weakness of the dataset is its size. We used a limited number of human annotators to label the event pair candidates, which was costly both in time and logistics. Crowdsourcing help obtain a much larger dataset.

Even though the ROCStories corpus has a moderate representation of everyday life, the events within it are not comprehensive as it does not cover every possible scenarios. However, we have not found a reliable methodology to extract high-quality event pair candidates from regular texts, such as news articles and Wikipedia pages. Such data sources would be worth pursuing for potential event pair candidates.

3.5.4 Classification

Because of the size of our training data, it is understandable that the classification methods we evaluated yield moderate results. With smaller training sets, deep learning models are prone to

overfit, which may affect their performances.

We have designed the Storyline network for the general purpose of inference on event pairs. However, there are limited instances in the EPOCI dataset where one event have multiple follow-up events from different story lines. Future work includes, the evaluations of the Storyline network on other event pair datasets.

3.6 Conclusion and Future Work

We proposed the EPOCI task, and provided a manually annotated dataset, for the evaluation of this task. We evaluated the performance of several methods for this task, as baselines for future models. This task is a good building block toward deeper understanding of the presuppositions and implications of events, which is necessary for reliable event inference. Future work includes building a much larger dataset of labeled event pairs, and the search for better computational models for the event ordering task.

Crowdsourcing could be an effective tool for obtaining a large amount of ordered event pairs. Crowd workers may not be committed to reliable performance, which may require careful management. Proper unsupervised methods could be leveraged with some relaxations. For example, instead of class labels of zeros and ones, we can consider using frequencies of ordered event pairs in a large corpus as the training targets, similar to contemporary research on temporal and causal relations between events.

The lack of proper training data is the challenge for computational models for the proposed task. Additional information, such as properties of the verbs, is a promising direction.

CHAPTER

4

STORY UNDERSTANDING BASED ON EVENT PAIRS

Stories in texts typically contain more than two events. Event inference on longer sequences of events is the basic requirement for the understanding of stories. Existing models on event inference have considered events as individual input points, without considering their pair-wise relations. In this research, we posit that inferentially related event pairs can facilitate the understanding of stories. As an effort to address $\mathbf{RQ_{infer\text{-}story}}$, we propose the development of PAIRS4STORIES, a method for event inference in stories based on information learned from inferentially related event pairs. We plan to investigate different strategies to utilize event pair information, compare their performance, and compete against existing event inference models.

4.1 Motivation

In certain types of stories, inferentially related event pairs are the key role players. In Chapter 2, we have shown that, in negative app reviews, user-reported user-app interaction stories consist of *user-action-app-problem* event pairs. Information residing in such pairs is the take-away lessons for developers who are trying to glean bug reports from app reviews.

In a previous study [10], we have conducted event inference on breach reports from the U.S. Department of Health and Human Services (HHS) [12]. A breach report tells a short story describing how a breach happened and the follow-up remedial actions taken by the responsible parties. These stories are comprised of *breach-description-remedial-action* event pairs. By predicting the events that may follow a breach description using event inference, our goal is to suggest security and privacy requirements for practitioners and end users that can be used to prevent and recover from such breaches. Table 4.1 shows the inferred remedial events based on a breach description, as well as what actually happened after the breach event. In a breach report, the *covered entity* (CE) is usually the responsible party that should take the remedial actions.

Table 4.1 Predicted remedial events for a breach description.

Type	**Sentence**
Breach Description	Two laptop computers with questionable encryption of individuals were stolen from the CE's premises.
Inferred remedial event 1	The CE filed a police report to recover the stolen item.
Inferred remedial event 2	The CE replaced its building alarm and installed bars on the windows.
Inferred remedial event 3	The CE revised its existing policies to ensure its vendors enforce appropriate security measures to protect ePHI.
Inferred remedial event 4	The CE implemented mandatory encryption for all mobile devices.
Actual follow-up event 1	The CE reported the theft to law enforcement.
Actual follow-up event 2	The CE worked with the local police to recover the laptops.
Actual follow-up event 3	The CE developed and implemented new policies and procedures to comply with the Security Rule.
Actual follow-up event 4	The CE placed an accounting of disclosures in the medical records of all affected individuals.

The inferential relations between event pairs, such as ((laptops being stolen), (reporting to law enforcement)) and ((laptops being stolen), (recovering the stolen item)), are the major propellers for the progression of the stories.

Finding inferential relations between events has practical significance in understanding simple stories, such as those in app reviews and breach reports. Relations between event pairs have been shown beneficial to event inference in stories [42, 45]. However, general story understanding require

the ability to infer events in longer sequences of evidence events. We posit that the information in different event pairs of a story can be combined to facilitate inference on longer event sequences.

For example, if we know the inferential relations in (*laptops being stolen*, *reporting to law enforcement*), (*laptops being stolen*, *recovering the stolen item*), and (*reporting to law enforcement*, *recovering the stolen item*), we can infer *recovering the stolen item* if the evidence events are *laptops being stolen* and *reporting to law enforcement*.

Therefore, we propose the development of PAIRS4STORIES, a method of conducting event inference in stories based on inferential relations between event pairs. We endeavor to solve event inference tasks, such as the Story Cloze Test, based on information learned from inferentially ordered event pairs.

4.2 Research Plan

Previous studies on event inference treat stories as is, i.e., sequences of atomic events, without the consideration of how the events are related to each other. We plan to treat a story as a graph of event pairs, where inferentially related event pairs are connected to contribute to the understanding of the entire story.

The tentative solution with this idea will be called PAIRS4STORIES. The current plan is to consider each event pair in a story as a node in a graph, and train a model with an attention layer that determines the importance of each pair in inferring the plausible held-out events. We will finalize the structure of the model, and then train and test the model on existing event inference tasks, such as event inference in breach reports and the Story Cloze Test.

To address $\mathbf{RQ_{infer\text{-}story}}$, we answer the following research questions when developing PAIRS4STORIES:

$\mathbf{RQ_{construct}}$ How do we effectively combine information of event pairs to facilitate event inference with longer evidence event sequences?

$\mathbf{RQ_{performance}}$ How effective is PAIRS4STORIES for event inference, compared to existing solutions?

To address $\mathbf{RQ_{construct}}$, we plan to investigate different structures of neural networks to combine knowledge from event pairs, and compare their performance. One baseline structure is to utilize only the last pair of events in a story, which has been proven to be moderately effective in story understanding.

To address $\mathbf{RQ_{performance}}$, we evaluate the performance of PAIRS4STORIES by reporting the accuracy, mean reciprocal rank, or other metrics of event inference tasks, and compare them with existing solutions that consider events individually instead of the interplay of event pairs.

BIBLIOGRAPHY

[1] Brandon Beamer and Roxana Girju. Using a bigram event model to predict causal potential. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing, pages 430–441, Berlin, Heidelberg, 2009. Springer Verlag.

[2] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[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. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[4] Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 789–797, Columbus, Ohio, June 2008. Association for Computer Linguistics.

[5] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. The spyware used in intimate partner violence. In *IEEE Symposium on Security and Privacy (SP)*, pages 441–458, Piscataway, NJ, USA, May 2018. IEEE Press.

[6] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE, pages 767–778, New York, NY, USA, May 2014. ACM.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Brussels, Belgium, June 2019. Association for Computational Linguistics.

[8] Venkatesh T. Dhinakaran, Raseshwari Pulle, Nirav Ajmeri, and Pradeep K. Murukannaiah. App review analysis via active learning: Reducing supervision effort without compromising

classification accuracy. In *IEEE 26th International Requirements Engineering Conference (RE)*, pages 170–181, Piscataway, NJ, USA, August 2018. IEEE Press.

[9] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 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)*, pages 499–510, Seattle, WA, USA, November 2016. ACM.

[10] Hui Guo, Özgür Kafalı, and Munindar P. Singh. Extraction of natural language requirements from breach reports using event inference. In *International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 22–28, Piscataway, NJ, USA, August 2018. IEEE Press.

[11] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. A needle in a haystack: What do twitter users say about software? In *IEEE 24th International Requirements Engineering Conference (RE)*, pages 96–105, Piscataway, NJ, USA, September 2016. IEEE Press.

[12] HHS Breach Portal. Notice to the Secretary of HHS breach of unsecured protected health information affecting 500 or more individuals, 2016. United States Department of Health and Human Services (HHS). `https://ocrportal.hhs.gov/ocr/breach/`.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, November 1997.

[14] Zhichao Hu, Elahe Rahimtoroghi, and Marilyn Walker. Inference of fine-grained event causality from blogs and films. In *Proceedings of the Events and Stories in the News Workshop*, pages 52–58, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[15] Zhizhao Hu and Marilyn A. Walker. Inferring narrative causality between event pairs in films. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 342–351, Saarbrücken, Germany, August 2017. Association for Computational Linguistics.

[16] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR)*, pages 41–44, May 2013.

[17] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, California, May 2015.

[18] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS, pages 3294–3302, Cambridge, MA, USA, December 2015. MIT Press.

[19] Andrew J. Ko, Michael J. Lee, Valentina Ferrari, Steven Ip, and Charlie Tran. A case study of post-deployment user feedback triage. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE, pages 1–8, New York, NY, USA, May 2011. Association for Computational Linguistics.

[20] Thomas Kober, Sander Bijl de Vroe, and Mark Steedman. Temporal and aspectual entailment. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 103–119, Gothenburg, Sweden, May 2019. Association for Computational Linguistics.

[21] Zijad Kurtanović and Walid Maalej. Mining user rationale from software reviews. In *IEEE 25th International Requirements Engineering Conference (RE)*, pages 61–70, Piscataway, NJ, USA, September 2017. IEEE Press.

[22] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC, pages 1009–1024, Berkeley, CA, USA, August 2015. USENIX Association.

[23] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *IEEE 23rd International Requirements Engineering Conference (RE)*, pages 116–125, Piscataway, NJ, USA, August 2015. IEEE Press.

[24] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 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*, pages 753–760, Stroudsburg, PA, USA, July 2006. Association for Computational Linguistics.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS, pages 3111–3119, Red Hook, NY, USA, December 2013. Curran Associates Inc.

[26] Seyed Abolghasem Mirroshandel and Gholamreza Ghassem-Sani. Towards unsupervised learning of temporal relations between events. *Journal of Artificial Intelligence Research*, 45(1): 125–163, September 2012.

[27] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

[28] Dennis Pagano and Bernd Bruegge. User involvement in software evolution practice: A case study. In *Proceedings of the International Conference on Software Engineering*, ICSE, pages 953–962, Piscataway, NJ, USA, May 2013. IEEE Press.

[29] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *21st IEEE International Requirements Engineering Conference (RE)*, pages 125–134, Piscataway, NJ, USA, July 2013. IEEE Press.

[30] Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 291–300, Piscataway, NJ, USA, September 2015. IEEE Press.

[31] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? Classifying user reviews for software maintenance and evolution. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 281–290, Piscataway, NJ, USA, September 2015. IEEE Press.

[32] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[33] Karl Pichotta and Raymond J. Mooney. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pages 2800–2806, Palo Alto, CA, USA, February 2016. AAAI Press.

[34] Karl Pichotta and Raymond J. Mooney. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 279–289, Berlin, Germany, August 2016. Association for Computational Linguistics.

[35] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[36] Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. Event2Mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 463–473, Melbourne, Australia, July 2018. Association for Computer Linguistics.

[37] Maria Riaz, Jonathan Stallings, Munindar P. Singh, John Slankas, and Laurie Williams. DIGS: A framework for discovering goals for security requirements engineering. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM, pages 35:1–35:10, New York, NY, USA, September 2016. ACM.

[38] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson Education Limited, 2016.

[39] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[40] Maarten Sap, Ronan J. LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. ATOMIC: An atlas of machine commonsense for if-then reasoning. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 3027–3035, Honolulu, Hawaii, USA, July 2019.

[41] Roger C. Schank and Robert P. Abelson. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum, Oxford, England, 1977.

[42] Siddarth Srinivasan, Richa Arora, and Mark Riedl. A simple and effective approach to the Story Cloze Test. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 92–96, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS, pages 3104–3112, Cambridge, MA, USA, December 2014. MIT Press.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS, pages 6000–6010. Curran Associates Inc., Red Hook, NY, USA, December 2017.

[45] Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. Integrating order information and event relation for script event prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–67, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[46] Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 3494–3501, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[47] Yinfei Yang and Amin Ahmad. Multilingual universal sentence encoder for semantic retrieval. `https://ai.googleblog.com/2019/07/multilingual-universal-sentence-encoder.html`, July 2019. Accessed: 2019-08-22.

[48] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 93–104, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.