CASPAR

Extracting and Synthesizing User Stories of Problems from App Reviews

Authors: Hui Guo, Munindar Singh

Problem Statement: App reviews contain a lot of valuable information that can be used to identify and rectify bugs in software applications. However not all reviews contain useful information that can be used for such a purpose. It can be very time consuming and expensive to use human labor to sift through the many thousands of reviews in order to extract this information. The authors of the paper attempted to use Natural Language Processing (NLP) and Machine Learning (ML) methods to address this problem.

Contribution: My contribution to the problem is creating my own implementation of the CASPAR method in Python using traditional libraries for NLP and ML in order to asses the effectiveness of the method. The implementation was successful and I was able to confirm and validate that the proposed method is very effective at addressing the problem.

Process:

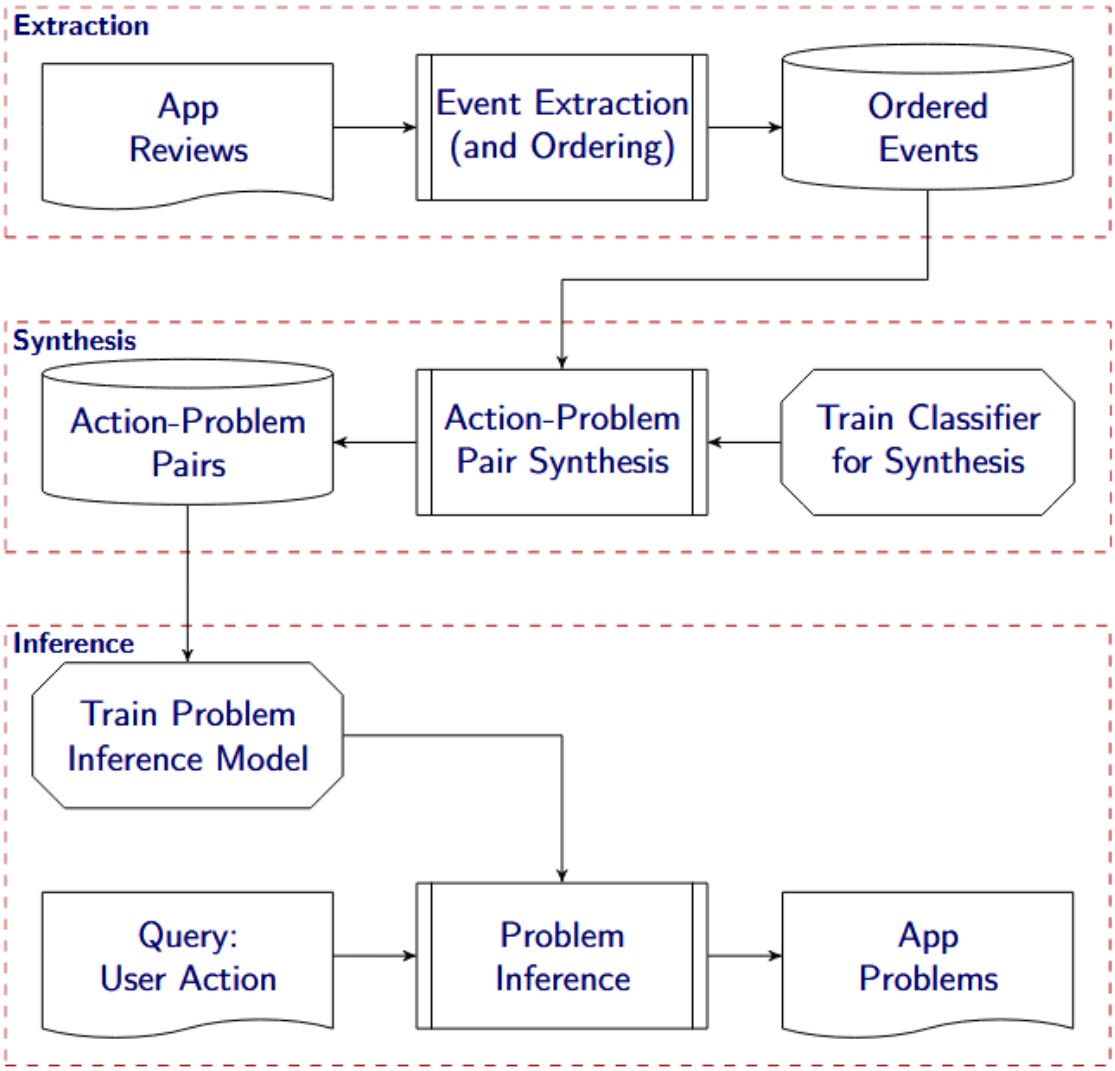

Figure 1: An overview of Caspar.

The method for automatic extraction of information from app reviews used by the Authors breaks down into 3 major steps

1. **Extraction**- The first step involves the extraction of what the authors call "key sentences". Key sentences are defined primarily via heuristics based on the presence of certain Key Words defined by the authors in the paper. NLP is then used to identify root verbs of sentences as well as verbs that are the root of an adverbial clause. These sentence fragments identified by the method are known in the paper as Events. These events are then ordered using more heuristics outlined in the paper in order to try to draw "cause -> effect" relationships between the events. When no heuristic can order the events then they are ordered by the order in which they are found in the review.

2. **Synthesis**- The second step is used to classify the events extracted in the previous step as either "User Actions", "App Problems", or "Neither". The idea is to try to correlate / connect "User Actions" with an associated "App Problem" in order to provide information about how the user found the bug so that developers can address the issue more quickly. Standard Support Vector Machine learning is used to achieve this. But first in order to convert the events into vectors that can be used by the SVM some type of encoding needs to be used. The authors found that the well established Universal Sentence Encoder (USE) works best. My results confirm this. The authors created a training set using Grad Students at the university to manually classify events. I used the same training set.

3. **Inference-** The third step is the inference step where we seek to infer what we can about the relationship between events. In other words we determine if our identified event pairs actually are related in a meaningful way. Given a pair of events (one user action and one app problem) we need to determine if the app problem is a valid follow up to the user action and not just a random event. To train the classifiers used in this step the authors used negative sampling to produce “random” events. Unfortunately I could not find the training set used by the authors for this step.