A Micromodule Approach for Building Real-Time Systems with Python-Based Models

Application to Early Risk Detection of Depression on Social Media

Rodrigo Martínez-Castaño[1], Juan C. Pichel[1], David E. Losada[1] and Fabio Crestani[2]

[1] Centro de Investigación en Tecnoloxías da Información (CiTIUS)
Universidade de Santiago de Compostela, Spain
{rodrigo.martinez, juancarlos.pichel, david.losada}@usc.es
[2] Faculty of Informatics, Università della Svizzera Italiana (USI), Lugano, Switzerland
fabio.crestani@usi.ch

Abstract. In this work we introduce Catenae, a new library whose main goal is to provide an easy-to-use solution for scalable real-time deployments with Python micromodules. To demonstrate its potential, we have developed an application that processes Social Media data and alerts about early signs of depression. The architecture has the following modules: (1) a crawler for extracting users and content, (2) a classifier pipeline that processes new user contents, (3) a RESTful API for alert management and access to users' submissions, and (4) a web interface.

1 Introduction

Early risk detection [2] is a challenging and increasingly important research area. People are exposed to a wide range of threats and risks, and many of them exteriorize on Social Media. Some risks might come from criminals and offenders (e.g., stalkers, or offenders with sexual motivations). Other risks are not originated by external actors, but by individuals themselves. For example, depression may lead to an eating disorder such as bulimia or anorexia or even to suicide. In this demo, we present the architecture of a system able to massively track online data and support risk assessment. The system is adaptable to multiple scenarios of early risk prediction, but we focus here on the case of depression. Such an application may be useful, for instance, to health agencies seeking a tool for analyzing the impact of depression on society.

Catenae is a Python library for building topologies in the shape of directed acyclic graphs. Graph nodes represent points of data transformation and edges symbolize data flowing between nodes. Nodes can be connected to multiple nodes both to send and receive data. The communication between nodes is managed in the form of message queues by Apache Kafka[3], a distributed message broker for

[3] https://kafka.apache.org/

high-throughput, low-latency handling of real-time data feeds. Each node can be instantiated and destroyed multiple times in such a way that if one type of node becomes a bottleneck, it is only necessary to replicate it. In addition, unlike popular batch processing frameworks where resources are assigned to the whole topology, Catenae assigns individually the hardware resources to each node.

We have developed a system for real-time prediction of signs of depression with Catenae. The system uses the social network Reddit as data source. Following the lessons learned in [1], we implemented a dynamic strategy that works with a *depression classifier* (built from the training split detailed in [1]) and incrementally analyzes the stream of texts written by each user. To meet this aim, we defined micromodules as tiny, loosely coupled software modules (nodes of the topology) that can scale horizontally and be deployed independently. The micromodule approach has some advantages over batch processing architectures when dealing with real-time independent events. This is the case in retrieving user texts (posts or comments) from Reddit in real time. Texts can be processed in parallel without any dependence among them. Our system is oriented to early detection and, thus, it is more reasonable to make the alerts as soon as there is evidence of a potential risk (rather than accumulating cases and making batch processing).

2 Early Risk Detection of Depression in Real Time

2.1 The Reddit Crawler

In order to maximize the number of tracked users, we have built a web crawler for Reddit following the rules expressed in the `robots.txt` file. The crawler also uses the Catenae library, as it is composed of multiple horizontal-scalable micromodules in a pipeline (see Figure 1):

- **Submissions crawlers.** They get all new submissions (posts) and extract their identifiers (as well as the user nicknames). Both types of identifiers are sent to different queues.
- **User crawlers.** They receive submission identifiers and try to extract as many users as possible. In Reddit, every submission has an author, a body of text and, possibly, some comments made by other users. Such thread of comments allows us to identify many other Reddit users. If submissions are old enough (i.e. have already many comments) then we can extract many user identifiers and send them to the queue of users for acceptance. Otherwise they will be relocated in a secondary queue with higher priority while this condition is not met. The goal of the age requirement is to find a balance between the number of collected users and the time spent in the queue.
- **New user filter.** Nicknames will be checked to avoid repeated users. Aerospike[4] is used to deal with this task efficiently. Those users who pass the filter will be sent to a queue of new users.

[4] https://www.aerospike.com

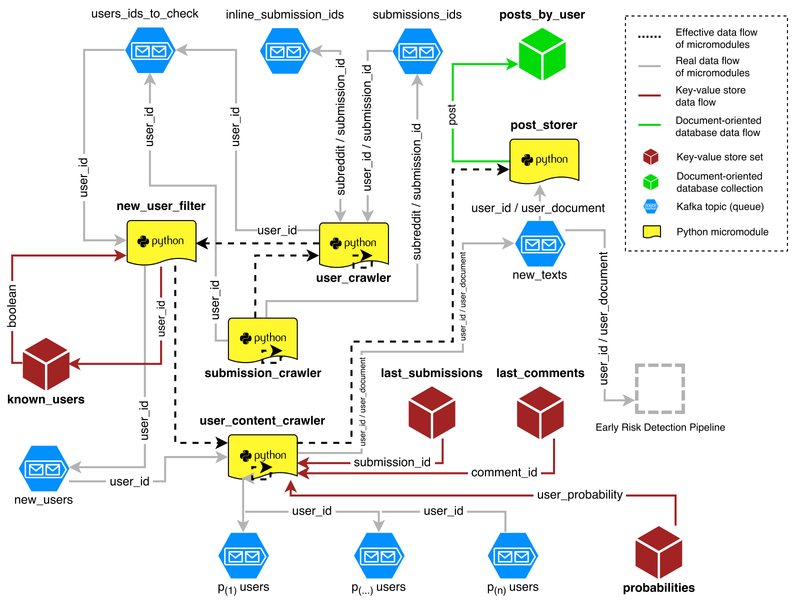

Fig. 1. Architecture diagram of the Reddit Crawler.

- **User content crawlers.** In this stage, all texts (submissions/comments) written on Reddit by the users that passed the filter are extracted as far as possible. Collecting all submissions of a user requires n calls, where n is the number of posts to retrieve (with a maximum of 100). On the other hand, retrieving the newest comments made by a user requires a single call. On every iteration, the system only retrieves the new texts available (it stores the identifiers of the last submission and comment for each user). The user content crawlers obtain user identifiers from different queues, ordered by priority. Based on the estimated probability of risk and the activity since the previous iteration, users can be relocated in different priority queues. A single output queue receives the extracted texts.
- **Post storer.** It is in charge of storing texts, grouped by user in a document-oriented database. In addition, these texts will also be fed to the Early Prediction Pipeline from the output queue of user content crawlers (and on-demand from the database through the RESTful API).

2.2 The Early Risk Detection Pipeline

The early prediction pipeline is a Logistic Regression classifier with L1 regularization, implemented in Python with *scikit-learn*[5]. The classifier is built with a training set of 486 users (83 positive, 403 negative) [1]. Each user is represented with a single document, consisting of the concatenation of all his writings. The prediction process has four micromodules (see Figure 2):

[5] http://scikit-learn.org

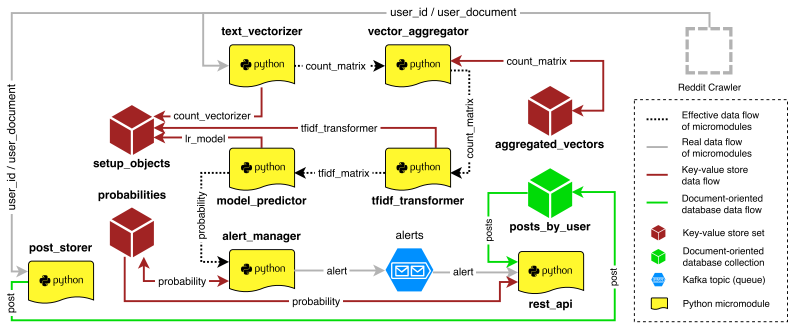

Fig. 2. Architecture diagram of the Early Prediction Pipeline. Main pipeline queues omitted for simplicity.

- **Text Vectorizer**. It transforms an input text into a vector of token counts.
- **Aggregator**. For each user, we accumulate a vector of token counts that represents all his texts. The aggregator merges the current vector of counts with the vector obtained from any new submission or comment.
- **Tf–idf Transformer**. It transforms the aggregated vector of counts to a normalized tf-idf representation.
- **Model predictor**. It produces the probability of risk of depression for a user given his tf–idf representation.

The aggregated vectors and the Python objects (count vectorizer, tf–idf transformer, and the classification model) are stored in the Aerospike store. In this way, the vector updates and the micromodules initializations are fast.

2.3 The Alert Manager API and User Interface

The user interface is a web application that retrieves information from the Alert Manager API. The Alert Manager RESTful API provides access to the generated alerts. Among its main functionalities are retrieving and processing those alerts, and processing the users' texts. Each alert has associated a confidence score (as produced by the classifier) and alerts are presented to the user by decreasing recency or in decreasing order of confidence. For each alert, the users' texts are presented ordered by decreasing probability of depression.

Acknowledgements

This work has been supported by MINECO (TIN2014-54565-JIN, TIN2015-64282-R), Xunta de Galicia (ED431G/08) and European Regional Development Fund.

References

1. D. Losada and F. Crestani. A test collection for research on depression and language use. In *Proc. of CLEF*, pages 28–39, 2016.
2. D. Losada, F. Crestani, and J. Parapar. eRISK 2017: CLEF lab on Early Risk Prediction on the Internet: Experimental foundations. In *Proc. of CLEF*, pages 346–360, 2017.