Twiagle: a Tool for engineering applications based on instant messaging over Twitter

Ángel Mora Segura, Juan de Lara, and Jesús Sánchez Cuadrado

Universidad Autónoma de Madrid (Spain)
{Angel.MoraS, Juan.deLara, Jesus.Sanchez.Cuadrado}@uam.es

Abstract. Microblogging services, like Twitter, are widely used for all kind of purposes, like organizing meetings, gathering preferences among friends, or contact community managers of companies or services.
With suitable automation, tweets can be used as a dialogue mechanism between users and computer applications, and we have built a tool, named Twiagle, to construct tweet-based applications. Twiagle includes a pattern-matching language to express the interesting parts to be detected and selected from tweets, and an action language to query matched tweets, aggregate information from them or synthesize messages.

1 Introduction

Microblogging and instant messaging systems are booming nowadays, thanks in part to the proliferation of smartphones and mobile devices. Services like Twitter[1] or WhatsApp[2] are extremely used nowadays to connect with friends, or to organize social activities. These services are not only used for leisure, but most companies and brands use these services to keep in contact with clients.

In this setting, we observe a growing need to automate social activities, leveraging on popular social network platforms, like Twitter. On the one hand, users of social networks – possibly lacking any programming skills – may wish to define simple applications involving the participation of a community of users. On the other, companies may like to open their information systems to social networks platforms, but this integration effort needs to be done by hand.

We claim that social networks based on instant messaging, in particular Twitter, are suitable as front-ends for computer-based applications. We call them *tweet-based* applications, and they present many advantages in some scenarios. First, instant messaging systems are designed to support a high load of users and messages, serving as a robust front-end, difficult to achieve for companies or end users. Second, many people are familiar with Twitter, and have it already installed. Hence, they do not need to learn a new application, or even install a new one. Third, applications can leverage from Twitter's social network structure.

We foresee three main kinds of scenarios for tweet-based applications. In the first one, Twitter is used as a front-end, which then needs to be connected to an

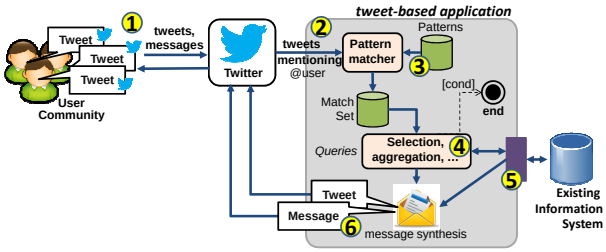
[1] http://www.twitter.com
[2] http://www.whatsapp.com/

existing information system (e.g., an airport may send notifications with flight information, or with status updates via Twitter to interested users). In the second scenario, small, simple, self-contained applications can be designed by unexperienced end users (e.g., outdoors educational games based on quizzes). Finally, an important scenario is the quick construction of applications to coordinate a large amount of people upon unexpected events, like natural disasters.

These scenarios present several challenges for this technology. First, if tweets are used as simple communication mechanism with the application, the relevant information needs to be extracted from them. Second, a mechanism is needed to specify simple actions, like querying the extracted information, or synthesizing messages. Finally, a quick, easy way for constructing this kind of applications is needed, enabling their use by non-experts, but supporting also their deployment into servers, and their integration with existing information systems.

This paper presents *Twiagle*, a tool for constructing tweet-based applications, including a Domain Specific Language (DSL) for expressing patterns, and a DSLs for describing actions. Sec. 2 describes our architecture for *Tweet-based applications*, Sec. 3 describes *Twiagle* using an example, and Sec. 4 ends with the conclusions and prospects for future work.

2 Architecture



The working scheme of our solution for tweet-based applications is shown in the inset figure, where the numbers illustrate a typical interaction. Firstly (label 1), users send tweets or private messages via Twitter. Then, the relevant information in tweets is extracted. Our solution relies on the definition of patterns, expected to be found in tweets. Not every tweet is sought, buy only those mentioning the user associated to the application, or private messages directed to it (label 2). The patterns (label 3) are defined by social media experts, or software engineers. A typical application may include different queries, selecting the relevant concepts in matching tweets, or calculating different aggregation values from them (label 4). In addition, data can be obtained or sent to existing information systems (label 5). The data extracted from queries, or provided by the information system can be used to synthesize tweets or private messages, directed to the users (label 6). Finally, conditions can be defined to signal the end of the execution.

In order to facilitate the construction of such system, we provide a Model-Driven Engineering solution, based on two domain-specific languages (DSLs). The first DSL (called *Twittern*) helps in the definition of relevant patterns, and concepts to be found in them. The latter are sets of relevant words, or fragments, and sets of synonyms can be automatically extracted from Wordnet [2]. The

second DSL, called *Twition*, is targeted to the description of the processing logic of tweet-based applications. It allows defining queries on tweets matching some pattern, using an SQL-like syntax. Queries can be used to select relevant information from tweets, or to calculate aggregated information from a set of tweets. The DSL also provides commands to synthesize private messages and tweets. Finally, it is also possible to define *data hooks*, a way to push extracted data into an existing information system, or to gather data from it. The next section describes a tool that realizes this approach.

3 Twiagle by Example

We show *Twiagle*'s capabilities through an example consisting in a simple voting among a set of users (see Fig. 1). The first step is to describe the interesting information in Tweets, using the *Twittern* DSL. A pattern is made of concepts, and in its simplest form, a concept is a set of words, which can be either defined explicitly by the designer, or can be automatically taken from a synonym set provided by Wordnet. We have also included specific Twitter concepts, like patterns to detect user names, URLs (specially pictures), and to define collections of interesting hashtags. The meta-data information present in tweets, like the originator, date or geoposition can be retrieved and does not need to be explicitly declared in patterns. Patterns also indicate if concepts have to appear in some specific order, or allow the interleaving of concepts with other words. It is also possible to specify that some concept cannot occur in a pattern, and whether concepts are to be sought ignoring upper/lower case, accents, and permitting missing vowels, as this is a usual idiom in tweets.

As a second step, our approach considers the description of actions by means of the *Twition* DSL. Twition allows issuing queries using an SQL-like syntax. They may refer to a set of matches of a pattern, as if they formed an SQL table, and the concepts in the pattern, as if they were SQL columns. Three kinds of specialized queries can be issued: Select (to select some concepts from a set of tweets matching a pattern), Adding (to perform some arithmetical operation on result sets), and Metadata queries (to obtain a result set made of some tweet metadata). Similar to data stream management systems [1] we may query using temporal windows. Currently, we support two kinds of temporal windows, one considering all data, and another one with the last tweet (@newest annotation). Once data becomes available from queries, messages can be composed and sent to a collection of users either publicly (command tweet), or in private, directed to a certain user (command message). In addition, received tweets can also be retweeted, and be categorized as favorite. Other commands include facilities to exchange data with an external source, and to signal the application end.

Each action has a name, so that actions can refer to the data they produce simply by that name. The type of data does not need to be declared, but it is inferred by simple rules. The execution model of Twition is based on data flow, relying on data dependencies, the recommended execution model for reactive, event-driven, scalable applications [3]. In this way, an action is performed as soon

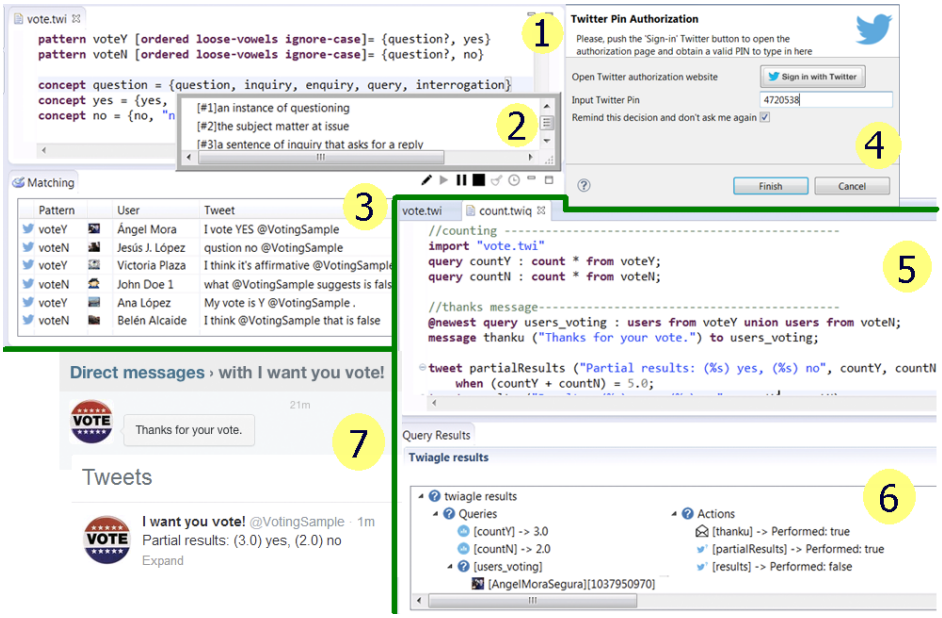


Fig. 1: (1) Defining patterns with Twittern, (2) Using Wordnet, (3) Testing with live tweets, (4) Authorizing Twiagle to use Twitter account data, (5) Defining actions with Twition, (6) Execution Debug, (7) Results shown in the Twitter console.

as its data becomes available, unless it contains an explicit trigger, in which case it is executed when the data is available *and* the trigger becomes true.

Twiagle includes a console to test patterns against live tweets, as well as an execution debug, showing the results of queries and actions performed. The tool is available at `http://www.miso.es/tools/twiagle.html`.

4 Conclusions and future work

In this paper, we have introduced *Twiagle*, a tool to build tweet-based applications. We are currently increasing the expressiveness of *Twittern*, improving *Twition* with new primitives, taking inspiration from data-stream systems for tweet querying. We are also working on the deployment mode, and considering support for other social networks, enabling inter-platform applications.

Acknowledgements. This work has been funded by the Spanish Ministry of Economy and Competitivity with project "Go Lite" (TIN2011-24139).

References

1. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS*, pages 1–16. ACM, 2002.
2. G. A. Miller. Wordnet: A lexical database for english. *CACM*, 38(11):39–41, 1995.
3. Reactive manifesto. `http://www.reactivemanifesto.org`.