Facilitating Twitter Data Analytics: Platform, Language and Functionality

Ke Tao*, Claudia Hauff*, Geert-Jan Houben*, Fabian Abel†, Guido Wachsmuth‡
*TU Delft, Web Information Systems, PO Box 5031, 2600 GA, Delft, the Netherlands
Email: {k.tao,c.hauff,g.j.p.m.houben}@tudelft.nl
†XING AG, Gänsemarkt 43, 20354, Hamburg, Germany
Email: fabian.abel@xing.com
‡TU Delft, Software Engineering, PO Box 5031, 2600 GA, Delft, the Netherlands
Email: g.h.wachsmuth@tudelft.nl

***Abstract*—Conducting analytics over data generated by Social Web portals such as Twitter is challenging, due to the volume, variety and velocity of the data. Commonly, adhoc pipelines are used that solve a particular use case. In this paper, we generalize across a range of typical Twitter-data use cases and determine a set of common characteristics. Based on this investigation, we present our Twitter Analytical Platform (TAP), a generic platform for conducting analytical tasks with Twitter data. The platform provides a domain-specific Twitter Analysis Language (TAL) as the interface to its functionality stack. TAL includes a set of analysis tools ranging from data collection and semantic enrichment, to machine learning. With these tools, it becomes possible to create and customize analytical workflows in TAL and build applications that make use of the analytics results. We showcase the applicability of our platform by building Twinder–a search engine for Twitter streams.**

I. Introduction

Since its launch in 2006, Twitter[1] has been attracting millions of users and media entities to share personal activities and publicize messages [1]. Given the immense amount of microblog posts published on Twitter every day, its popularity makes it an attractive source for conducting large-scale data analytics. In the era of "Big Data" with emphasis on the 5 V's (Volume, Velocity, Variety, Value and Veracity), the characteristics of *Volume* and *Velocity* in Twitter data analytics are represented by the hundreds of millions of Twitter messages posted every day and the TPS (Tweets per second) record broken as important events are talked up with microblog posts[2]. The *Variety* comes both from combining the textual messages (limited to 140 character) with the metadata and the need of integrating external knowledge (e.g. knowledge bases). The final two 2 V's, *Value* and *Veracity*, cannot be naturally obtained from Twitter data - human insights and ideas drive those two dimensions. Thus, deriving valuable and high-quality insights from Twitter data become non-trivial challenges.

The openness of Twitter has attracted numerous researchers to conduct analytics on various scenarios, ranging from sport events [2], [3] and natural disasters [4] to political elections [5], [6] and users' cultural characteristics [7]. Previous works [8], [9], [10] in the context of information retrieval for Twitter data focused on fulfilling the general information need from users with a list of ranked search results, enhanced by analytical results, including frameworks for relevance estimation and duplicate detection. Besides the scientific contributions, applications have also been developed based on analytical results obtained from Twitter data. For instance, Sakaki et al. [11] have established an early warning system for earthquakes in Japan; later a similar system was also established in the United States Geological Survey [12]. Gao et al. [25] proposed a Twitter-based user modelling framework, which developers can leverage to build their personalized applications. Abel et al. introduced Twitcident[3] [13], a framework to fulfil the information needs from (semi-)public sectors by filtering Twitter streams, making use of the semantics in tweets, and data analysis techniques. In order to provide a systematic solution to content analysis tasks, IBM created the framework UIMA [14], which later became an Apache project, to analyze unstructured information with the aim of providing relevant knowledge to end users. However, there is to our knowledge not yet a dedicated solution for Twitter (or more generally microblog-based) data analytics that (i) allows for efficient **customization** of the tasks, (ii) with an **extensible** set of functionality, (iii) which can be employed both for **research and application development** purposes.

In this paper, we tackle this challenge and introduce *TAP* (*T*witter *A*nalytical *P*latform), a data analytics platform for Social Web and Twitter data in particular. TAP features a stack of analysis tools, which can be chained into customized workflows. Depending on the analytical tasks, the workflow, which may be programmed via a domain-specific language TAL (*T*witter *A*nalysis *L*anguage), can handle Twitter data in either a streaming or a batch manner. We make the platform open-source and publicly available[4]. To showcase the ability of supporting the efficient application development with the analytical results provided by TAP, in this paper

[1]http://twitter.com/
[2]https://blog.twitter.com/2013/new-tweets-per-second-record-and-how
[3]http://www.twitcident.org/
[4]https://github.com/ktao/tap

we present the implementation of our *Twinder* prototype–a search engine for Twitter streams[5].

The main contributions of this paper are:

- We analyze typical use cases of Social Web data analytics and abstract the procedures in common into a conceptual pipeline, which serves as the fundamental building blocks of our Twitter analytics platform (Section III).
- We present *TAP* (*T*witter *A*nalytical *P*latform), a data analytics platform for Twitter that features support for both streaming and batch processing of Twitter data (Section IV) with scalable infrastructures.
- We introduce the domain-specific language *TAL* (*T*witter *A*nalysis *L*anguage), which allows us to utilize TAP's set of analysis tools and in turn enables us to create and customize analytic workflows in a simple manner (Section V and VI).
- We demonstrate TAP's ability to perform data analytics tasks and support applications through an in-depth analysis of different implemented features aimed at the improvement of retrieval effectiveness in adhoc search settings (Section VII).

II. Background: Social Web Data Analytics

Nowadays, the data being generated on Social Web applications including Twitter, Facebook, Flickr etc., is of massive volume. The popularity of these services makes it possible to turn users into "Social Sensors" [11] for conducting analytics in different application scenarios, including commercial marketing, recommendation systems [15], [25], political elections [16], and, public infrastructures [17], [13]. However, one has to understand the characteristics of Social Web data before such kind of analytics can be implemented. The objective of Social Web Data Analytics is to provide consumers with "knowledge to act", in order to help them to make correct decisions. These decisions, in commercial cases, can yield profits or avoid losses. In this section, we will present a number of Social Web data characteristics, that make data analytics challenging. Then we move on to showing how to tackle these challenges within TAP and TAL.

As the content on the Web is created in a collaborative manner, a large number of applications and Social Web portals allow users to produce, consume and edit content as well as to vote and comment on other users' content. Thus, Social Web documents (of any type, e.g. image, video, audio, microblog message, blog post) are "rich" resources, with user-generated meta-data and signals allowing us to perform complex analytics.

Size and Ownership Social Web portals have been growing quickly and often are continuing to grow; they may be generating hundreds of millions of items per day. For example, on an average day, 500 million tweets are being posted on Twitter, 1.6 million public photos are uploaded to Flickr and 500 terabytes of data are ingested into the Facebook database.

Not only size and growth make data collection difficult, the fact that most Social Web portals allow users and developers only very limited access to the data adds another dimension to the problem.

Unstructured Format Before knowledge can be extracted from Social Web data, a lot of effort has to be expended on the refinement and transformation of the data [18]. This is not a simple process as multiple software tools may be involved in the analytics pipeline. Previous studies [19], [20], [15], [21] have shown that more insights can be obtained by enriching the Social Web content.

Data Noise The simplicity of authoring and the fact that large financial incentives exist for adversaries (to produce spam) lead to a substantial fraction of Social Web content to be of low quality. Given the collected data, one of the key challenges is to filter out the noise. Filtering can be based on simple manually defined rules, or rather complex NLP-based techniques.

III. Social Web Data Analytics Pipeline

Having analyzed the characteristics of Social Web Data, we now propose a Social Web Data Analytics pipeline in four steps: the analytics tasks follow a process of (i) collecting data, (ii) filtering the data, (iii) enriching the data with knowledge from other sources, and (iv) mining the refined data.

In order to orchestrate these different steps, they need to be connected in a pipeline. Additionally, the data needs to be translated into one generic data model. After the application of the analytics pipeline, the results can then be exploited to support secondary applications, such as interpreting the data through visualizations [2] or providing public sectors with real-time information during emergency situations [13]. Moreover, by integrating different components into our analytics pipeline, we can connect them together to make the analytical tasks more complex and more powerful.

To provide the reader with a better intuition of the proposed steps and their integration, we now discuss the major pipeline components in the context of three Twitter-based use cases:

User Profiling Previous works showed the feasibility to leverage Social Web activities for user modelling and personalization [25]. User profiles can be build from a user's semantically enriched tweets [19]. These user profiles in turn can then be employed for applications such as news recommendation [15].

Crisis Management Twitcident [13] is an application which provides interested parties (e.g. the police, the council) with Social Web information support during emergency

[5]The demonstration, the detailed information, and the latest development progress about of can be found at http://twinder.github.io

circumstances. Twitcident connects to emergency broadcasting services and automatically starts tracking and filtering incidents when an "interesting"' event is identified.

Brand-building Major corporations have realized the importance of brand-building on Social Web applications. Apart from proactively engaging with customers through multiple channels, it is necessary to monitor the public sentiment towards their products or services and to react accordingly [22].

Therefore, it would be useful if one can quickly build an application for monitoring the Social Web data streams and automatically categorize the information into different priority levels so that Public Relation efforts can be spend in a optimal way.

In the rest of this section, we will describe how these three cases fit into our Social Web Data Analytics pipeline.

A. Data Collection

Let us now discuss data collection issues within the context of the three introduced use cases.

1) User Profiling: When new users begin to use the application supported by the User Profiling module, their historical activities should be acquired from Twitter to build a model that is as accurate and complete as possible. As users keep using the application, we also need to monitor their latest activities as interests may vary over time [23].

2) Crisis Management: Twitcident is connected to an emergency broadcasting system that can provide basic information about an incident. This may include several relevant keywords, the incident type, and possibly a geo-location. There are two ways to collect tweets that are potentially relevant to the incident:

- We can monitor the keywords that describe the incident and its type and the physical area around the incident (identified through geo-coordinates).
- With full access to historical data, we can build an index of Twitter messages' content; the potentially relevant tweets can be acquired by issuing a query against the index.

Having acquired the original tweets that may be relevant to the incident, we can further analyze their likelihood of relevance and assign them to different facets or categories.

3) Brand-Building: To monitor the tweets discussing a certain produce or service, we can use its name as a keyword or simply follow the account of the brand (including replies & direct tweets to this account). However, the tweets acquired may be much more than what want for brand-building purposes. Here the next phase of the pipeline becomes important: *data filtering*.

B. Filtering Social Web Data

The filtering of messages is an important component in a number of analytical tasks:

1) User Profiling: During the interest-based profiling of a user, we usually track the user by her ID via the Twitter Streaming API (see Section VI-A). This process however will not only provide us with her posting activities but will include those messages that mention her. Here, filtering is simple: we retain all messages that are authored by the given user and remove the remainder from the stream.

2) Crisis Management: Twitcident provides users with information derived from Twitter on multiple aspects of an incident. We can imagine that during incidents which receive a lot of attention some tweets may be retweeted frequently. However, from the informativeness point of view, these retweets add little value for handling the incident. Therefore, we can filter them out.

3) Brand-Building: Many companies assign cool names to their produces or services, but frequently they may be easily confused with other entities with the same name. For example, Microsoft may want to know the public opinions about their Windows products after a new version release. However, "Windows" can refer to the operating systems distributed by Microsoft or a range of other concepts. By keyword matching, we collect all tweets mentioning "Windows" independent of the underlying semantics. To tackle this problem, we can use semantics extraction tools (see Section III-C) to identify the concepts in the tweets. Then we can rely on our filtering component to only retain the tweets that discuss the wanted concept.

C. Enriching Social Web Data

As Social Web data is often unstructured and superficial (see Section II), we require approaches that enrich the data with further evidences (semantics). We can use existing techniques, including natural language processing, knowledge bases, Semantic Web resources etc., to extract valuable meta-data automatically. However, the specific method depends on the type and the characteristics of the data. In some cases, the results need a further normalization step before they can be used for analytics (in particular numerical data with units attached often requires normalization).

Unstructured textual raw data is difficult to exploit for analytics in case of Twitter, due to the severe length restrictions and the informal nature of most messages. With semantics identification tools, we can make better use of textual data by linking the concepts to a structured knowledge base. Aggregation tools can add more summary results for complex data, especially from lists.

We now discuss the enrichment process with our three use cases in mind:

1) User Profiling: Given the Twitter messages' content posted by some user, we can extract named entities and topics to better understand the semantics of her Twitter activities. For this purpose, we utilize Named Entity Recognition (NER) tools such as OpenCalais[6], which have been shown

[6]http://www.opencalais.com

to work well for microblog messages [24]. This leads to semantically enriched documents whose identified semantic concepts form the basis for the user profile.

2) Crisis Management: In response to a crisis, Twitter users talk about different aspects such as the reasons, locations, damages, casualties, etc. The identified concepts can be used to organize information facets and allow relevant parties to quickly zoom into the aspect that is most pertinent to them.

3) Brand-Building: Customers may talk about their experiences of not only using products, but also about purchasing, delivery and after-sale services. In large companies these different types of messages should be categorized according to the message type, the country of origin, the type of user, etc. Enriching tweets with semantics and identifying the related concepts in contents and metadata may help in this procedure. Moreover, brands may want to prioritize the messages to first review and respond to messages with a negative sentiment.

D. Mining Social Web Data

The core challenge of Social Web Data Analytics is to extract "knowledge to act", or the information that can instruct or support the business in real life. Data mining techniques are primarily designed to handle large-scale data, extract actionable knowledge, and gain insightful results. Therefore, we consider "Mining Social Web Data" as the last phase of Social Web Data Analytics. The data collection, filtering, as well as enrichment can be considered as the preparation for the eventual mining process.

1) User Profiling: User profiles can be derived without a specific application in mind [25]. However, they are most useful in practice, when employed for a specific task, such as advertisement targeting. In this example, the user profile is used as input to a classification model which determines whether or not to show a particular ad to the user.

2) Crisis Management: The tweets discussing an incident may provide information on different aspects. One can either manually define rules for categorization, or use classification algorithms to achieve the same. The latter is the only feasible approach for large-scale data sources, as manual rules can never capture all particularities of unstructured documents.

3) Brand-Building: The complaints from a user may have effects on a product's or service's reputation, depending on her influences on followers and the attractiveness of her messages. Therefore, it would be effective if the complaints can be categorized (classified) with respect to their priorities.

Having introduced the three use cases and analyzed the detailed requirements in each of the four steps in the Social Web Data Analytics pipeline proposed, we are going to provide a more concrete and complete solution in the rest of this paper. We implement the 4-step pipeline model in the form of *workflows* that can be customized with the domain specific language and enabled by a set of tools.

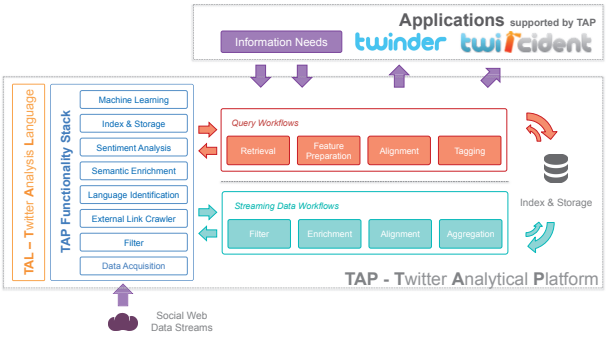


Figure 1. Architecture of Twitter Analytical Platform

IV. Twitter Analytical Platform

Having analyzed the characteristics and key enabling technologies, we will now provide our systematic solution for conducting data analytics of Twitter data. Towards this end, we have designed TAP (the **T**witter **A**nalytical **P**latform), which allows us to develop customized analytical workflows with Twitter data. Our platform is open-source[7] and can be easily extended.

A. Architecture

The architecture of TAP is summarized in Figure 1. The analytical tasks are implemented as workflows that can be executed on the platform. The workflows rely on the tools provided in the *TAP Functionality Stack*, which features data collection, filtering, enrichment, and mining capabilities. The workflows can be programmed in the domain-specific language TAL (**T**witter **A**nalysis **L**anguage), whose data model and syntax will be described in Section V. With this language, one can select a set of analysis tools in the *TAP functionality stack* as we will discuss in Section VI. The currently supported analysis tools in the TAP functionality stack are listed in the blue block in Figure 1. We have designed a unified interface for these tools.

The intermediate results and historical data can be indexed and stored in an internal facility, which is depicted by the grey component in Figure 1. Currently, TAP relies on typical tools of indexing and schema-free databases, which achieve good scalability, so that the Twitter data of large *Volume* and high *Velocity* can be handled – it provides the solution to the *Size* problem that we identified in Section II.

B. Workflow Design

Our platform provides users with the freedom to create and customize their own analytical workflows. The workflows can be categorized into two types: (i) streaming workflows, and, (ii) query workflows. The categorization depends on the length of the acquired Twitter data (infinite

[7]https://github.com/ktao/tap/wiki

or finite). Typical applications are supported by TAP in a hybrid mode, i.e. they rely on both types of workflows with various purposes. However, one can, of course, solely use one of them according to the requirements of the envisioned analytical task.

1) Streaming Workflow: The streaming workflows start with obtaining a Twitter data source from which Twitter messages arrive continuously. The workflows of this type are typically used for pre-processing the data collected from Twitter.

Here, some of the analysis tools can already be applied to the tweets before a specific information need is specified. These pre-computed intermediate results are stored in (and later served from) the internal storage.

For instance, in the use case of user profiling, one can implement a user modelling service with TAP by following the methodologies from our previous works [26]. The semantic enrichment of user Twitter activities [19] can be implemented with a streaming workflow in TAP and the enriched Twitter messages will be indexed and stored for later user profiling, which can be implemented as a query workflow.

2) Query Workflow: The query workflows serve the information needs specified by users, e.g. a keyword search query, the sentiment over a brand or a new product to be monitored. The preprocessed (i.e. filtered and semantically enriched) Twitter messages can be fetched from the internal index & storage facility for data mining purposes. The necessary features are either fetched (if pre-computed) or generated on the fly and served to the selected learnt machine learning algorithms.

Following our user profiling example, a query workflow can be designed to compute her user profile. We collect her streamed activities and build the user profile with the user modelling strategy proposed in [25]. However, even if we have not been monitoring the user, we can still collect the user activities from external services and apply the same preprocessing tools in the streaming workflow. It should be noted that there is no restriction on the functionality depending on the workflow type. However, in this case, the efficiency of the user profiling service may be lower as the high-quality user profiles rely on the results from a chain of analysis tools.

V. Twitter Analysis Language

Having introduced the architecture and the workflow design, we now present a domain specific language TAL (*T*witter *A*nalysis *L*anguage), with which we can program the *TAL scripts* for customizing the various analytical workflows. The language will provide an interface for building the analytics workflows efficiently. In this section, we introduce the essentials of this language, including the data model and the syntax.

Note, that latest specification and example usages can be found at the development page of our Twitter Analytical Platform[8].

A. Data Model

TAP provides a unified data model to accommodate the source data, the intermediate results, and the output. Therefore it tackle the problem of *Unstructured Format* identified in Section II. Given the fact that TAL is focused on data analytics with tweets, the core element in the data model is a single Twitter message. Thus, if we denote a single tweet as t with a subscript, the data model can be represented as follows.

$$t_1, t_2, t_3, \ldots, t_n, \ldots \tag{1}$$

as stream or

$$t_1, t_2, t_3, \ldots, t_n \tag{2}$$

as finite list.

The tweets arrive in the order indicated by the subscript. For both workflow types, one can utilize the available tools discussed in the TAP Functionality Stack to conduct data analytics.

The core element of the data model in TAL is the representation of a tweet. It has numerous attributes which can either be directly received from Twitter or derived from existing attributes through external services. The attributes can be either a value, a nested value, or a list of values. For example, a tweet that is received from the Twitter Streaming API looks as follows:

```
{"t1": {
    "text":"Pageview logs of Wikipedia are publicly
         available at http://t.co/WD9hNUmL5z , must be
         useful for some analysis. #RAMSS2013 #WWW2013",
    "source": "web",
    "author": {
        "username": "taubau",
        "id": 17730501,
        "created_at":"Sat Nov 29 07:47:38 +0000 2008",
        "statuses_count":797,
        "friends_count":369,
        "followers_count":160 },
    "created_at": "Tue May 14 19:16:20 +0000 2013",
    "id": "334386718419587100" }
    "hashtags":["RAMSS2013","WWW2013"],
    "language":"en",
    "urls":{ "http://t.co/WD9hNUmL5z":s
      "http://dumps.wikimedia.org/other/pagecounts-raw/"},
    ... (more attributes)},
... (more tweets),
 "meta": {
  "started_at": "Tue May 1 00:00:00 +0000 2013",
  ... (more meta information)}
}
```

Every single tweet in either the stream or the list is supposed to be a *tweet element*, as shown in the above data model. Besides the tweet elements, there is also the *meta element*, which contains a summary or global information about the whole data stream or list. This data model can accommodate

[8]https://github.com/ktao/tap

various operations on the data acquired from supported sources, including filtering, enriching, and mining tools provided by the TAP functionality stack.

In TAL, each attribute has a data type, which can be one of the three supported types: (i) numeric, (ii) boolean, and (iii) string values. The numeric value can be integer or double values. Furthermore, the calculation can be performed between attributes or with immediate values of these three types. The supported operators are described next.

B. Syntax

The TAL scripts contain a series of *statements*. Each of them can specify an operation, such as collecting data from sources, making changes to the attributes of tweet elements or the meta element. There are two categories of statements in TAL:

General Operation The statement of *General Operation* type is for the overall operations to the target of analytics, including data collection, indexing, storage.

Assignment The *Assignment* statements can create or modify the attributes of elements, including both tweet elements and the meta element.

When writing scripts in TAL, the keyword *this* always refers to the data that is currently being processed. The *General Operation* statements can define the data source or invoke the indexing as well as the storing procedure as follows:

```
[General Operation](parameters)
```

Depending on the specific *General Operation*, certain parameters can be passed. For instance, one can specify the data source to be analyzed as tracking the keyword "twitter" via the Twitter Streaming API with the following statement (see detailed usage in Section VI-A).

```
source.twitter.filter("twitter", null, null)
```

One can use the keyword *this* with a dot operator (.) to specify the attribute of all the tweet elements, or with an arrow operator ($\rightarrow$) to cite a particular element, especially the meta element. Given the method of specifying an attribute in an element, the general syntax of assignment operations is:

```
this.[attribute] := [method](parameters)
this->meta.[meta attribute] := "example"
```

The statements above describe two assignment operations: (i) to assign the specified *attribute* with the value derived by the *method* with required *parameters* (if applicable) and (ii) to set the value of a meta attribute to the string value "example".

1) Operators: TAL provides support of expressions with a set of operators. Depending on the data types or purposes, different sets of operators can be used.

Boolean Values The logical operators supported by TAL are ! (NOT), AND, OR. Thus the logical expressions can be connected to become a complex logical expression.

Numeric Values Besides the arithmetic operators, e.g. $+$, $-$, $*$, $/$, we can use the relational operators, including $==$, $!=$, $>$, $<$, $>=$, and $<=$, to determine a boolean value.

String Values TAL provides the following operators for the data type of String to formulate logical expressions: $==$ (equal), $!=$ (not equal), $contains$, $startsWith$, $endsWith$. The length of a string value can be calculated by using the operator len.

Aggregation A number of aggregation operators are currently being supported by TAL, including $minimum$, $maximum$, $median$, $average$, sum, $count$ and $cardinality$. The first five operators can only be used on numeric attributes, while the operators $count$ and $cardinality$ are used for counting the elements in a certain attribute or the distinct values for the attribute. Moreover, the aggregation operator can also be combined with a condition so that we can derive statistics of a subset of elements without removal of data.

Miscellaneous TAL provides built-in support for many operators that can be used in expressions, including (i) $overlap$ for determining the overlap between two lists with given identifiers and (ii) $exists$ for checking whether the given attribute exists for this tweet.

Delete One can remove the intermediate results or unnecessary content from the data with the $delete$ operator, especially before moving the results in streaming workflows into the internal index and storage. This in turn reduces the spatial costs.

2) Analysis Functions: TAL relies on the analysis tools in the *TAP Functionality Stack* to construct or derive evidences for conducting analytics. These tools can be invoked from within TAL; detailed information on currently supported analysis tools can be found in Section VI.

C. Implementation

To implement the Twitter Analysis Language, we employ the *Spoofax Language Workbench* [27], a platform for the development of textual domain-specific languages with state-of-the-art IDE support. Spoofax provides highly declarative meta-languages, which abstract over the implementation of language processors. This allows us to focus solely on the design of TAL. In particular, we provide a modular syntax definition of TAL in SDF3, name binding and typing rules in NaBL and TS [28], and a mapping to Java code in Stratego [29]. Given these declarative specifications, Spoofax derives a full-featured Eclipse editor plugin [30] for designing workflows in TAL and generating Java code from them. The generated Java code can be executed in TAP. We make the latest progress on the implementation of TAL publicly available[9].

[9]https://github.com/guwac/metaborg-tal

VI. TAP Functionality Stack

As mentioned in previous sections, the analytics workflows depend heavily on the analysis tools supported by TAP. In this section, we are going to introduce each of them with example usage in TAL.

A. Data Collection

The first step of creating a workflow is to acquire the data to be analyzed. As mentioned in Section IV-B, TAL supports processing both data streams and a tweet list of finite lengths. TAP provides access to the Twitter streaming API[10], the retrieval interface of an internal index and storage infrastructure, and is able to obtain previous tweets from the Twitter REST API[11].

1) Streaming Data: TAL supports acquiring real-time data from a public stream endpoint of the Twitter Streaming API. Specifically, users can use either a *filter* endpoint with parameters or a *sample* endpoint. Depending on the endpoint selected, the streaming data can be acquired with the general statement "twitter.[endpoint]". For instance, the following example specifies that we need to monitor the tweets either mentioning a keyword **twitter** or tagged with a geo-location within New York City.

```
twitter.filter("track=twitter&locations=-74,40,-73,41")
```

2) Search Result: Besides streaming data, a tweet list of finite length is often used as source to fulfil a specific information need. The most common practice is to take advantage of information retrieval tools to generate a ranked list of search results. As mentioned in Section IV-A, TAL maintains an internal index so that one can search with keyword queries. In order to obtain the search result list, one can fill the query as a parameter in the statement starting with the keyword *search*.

TAL can accept the search query that follows the query language specification of the preferred index. For instance, if an Indri search engine is deployed as the indexing component in TAP, one can specify the search query with the Indri query language. Besides the original information given by Twitter, the search result given by TAL will not only provide all the pre-processing results in the persistent layer but also attach the query along with the retrieval score to the tweet. Therefore, a storage facility is needed to achieve this. A hybrid solution would be to explicitly configure the search engine to store the source information. An example usage of this function is shown below:

```
search([search query])
```

3) Previous Tweets: Apart from the index, TAL maintains an internal storage for previous tweets. In this way, it allows us to retrieve a selection of stored tweets. For example, we can retrieve the tweets posted by a specific user, given her Twitter user ID:

```
twitter.select(author.id = 17730501)
```

B. Filter

With acquired data of either continuously arriving streams or lists of finite length, the selection effectiveness is limited to the functionality provided by the sources therefore further refinement is still needed for analysis in many cases. The most basic approach to achieve this is to filter on the attributes of tweets. For example, we can remove the non-English tweets if we focus on analyzing the English tweets as we did in some previous work [8], [10]. To achieve this in TAL, one can utilize the *filter* function, with a logical expression that can be evaluated on each tweet:

```
filter([logical expression])
```

The filter method will iterate over all the tweets and evaluate the logical expression. Only the tweets with an evaluation result of *TRUE* will be retained. We can leverage this function to deal with *data noise* problem that we have noticed in Section II. More evidences on which one can perform the filtering can be derived from other functions introduced in this section.

C. External Link Crawler

Due to the length limitation of 140 characters, tweets are too short to accommodate the full stories so that frequently URLs are included. TAP provides support for the extraction of the main content in the resources referenced by these URLs. Moreover, the URLs in the tweets are often shortened by various services. We implement two functions to address these issues: *URL Expansion* and *Web Content Extraction*. Note that Twitter currently processes all URLs with the *t.co* URL wrapper and includes the "expanded URLs" in the URL entities of tweets. However, it will not expand the URLs shortened by a third-party service.

1) URL Expansion: Assuming that a list of URL entities are given in the field "urls" in every tweet, they can be expanded as follows:

```
this.urls.expandedurls := urlexpand(this.urls)
```

The URL expander will iterate over the URL entities in the list as specified by the attribute "urls" in every tweet and attempt to follow the links until it receives an *HTTP Success* response. The expanded URLs will be stored in the new attribute named "expandedurls" in every element of the list "urls".

2) Web Content Extraction: Previous studies have shown the effectiveness of exploiting the linkage for inferring semantics from tweets [19]. The Web Content Extraction function allows for extracting the main content that the URLs refer to. To use this function, we need to specify a list of URLs. As the result, a list of URL entities, including an

[10]https://dev.twitter.com/docs/api/streaming
[11]https://dev.twitter.com/docs/api/1.1

attribute as specified for storing the result, will be returned. For example:

```
this.urls.content := extcrawl(this.urls)
```

The results will be added into the corresponding URL entities in the list as attribute *content*.

D. Language Identification

Given the text in the attribute, its language can be identified and a new attribute will be added to store the language identifier. As of March 2013, the language identification of original Twitter messages are provided on streaming API[12]. However, we reserve the functionality in our design since it can be applied to other attributes as well.

For instance, the following command identifies the main language used in the *text* attribute; the results are stored in the *lang* attribute:

```
this.lang := enrich.langid(this.text)
```

E. Semantic Enrichment

Previous work has shown that semantics are meaningful for various analytical tasks, including understanding the user preferences [15], relevance of tweets to given topics [8], [9], and serving the modelling interface for applications such as recommender systems and digital tour guides [31]. TAP provides access to services like OpenCalais (oc), DBpedia Spotlight [13] (dbp), and Wikipedia Miner [14] (wm), which can annotate the named entities mentioned in the input text. The results, in the form of a list of named entities, can be attached to the analysis object with the given attribute name. For example,

```
this.semantics := semantics.dbp(this.text)
```

The statement above calls the DBpedia Spotlight service. The outcomes will be stored as a new attribute *semantics*. Moreover, semantic enrichment can also be applied to the external resources. This can be realized by specifying the attribute of the content crawled by the External Link Crawler.

F. Sentiment Analysis

TAP provides built-in support for categorizing the sentiment of Twitter messages. For example:

```
this.sentiment := sentiment(this.text)
```

The results of the sentiment analysis will be attached to the tweets as a new field. The result value can be: *positive*, *negative*, or *neutral*.

[12]https://dev.twitter.com/docs/platform-objects/tweets

[13]http://spotlight.dbpedia.org/

[14]http://wikipedia-miner.cms.waikato.ac.nz

G. Index & Storage

TAP provides an internal index and storage facility, which allows us to retrieve tweets based on keyword or more complex queries that follow the given query language specifications. For example, the following statement asks TAP to store the tweets and add them into index:

```
store()
index()
```

The internal index and storage facilities are commonly used to preserve the preprocessed information that can be used for analytics at a later usage (again). This together with *Data Collection* function (see Section VI-A) thus provides a solution to the *Ownership* problem that we have identified in Section II.

H. Machine Learning

In existing works, researchers have been applying various machine learning algorithms for analytical tasks. TAP relies on Weka tools [15] to provide support for a wide range of classification and clustering algorithms.

1) Classification: Classification algorithms are supervised learning methods. Hence, we need to specify the model derived from training data. Additionally, an attribute-feature mapping file that describes the relationship between the attributes in the TAP data model and the features in the learning model needs to be provided. Simple normalization operations can also be defined in the mapping file. Consider the following TAL example to this effect:

```
this.relevance := ml.classify(this,[MODEL],[MAPPING]}
```

The above statement was taken from a TAL script we employ within our relevance estimation workflow. The workflow estimates the relevance of tweets to a given topic (query). Assuming that the candidate tweets are readied for the application of the relevance estimation model, e.g. all evidences are included as attributes, one can pass the classification model file name as well as the attribute-feature mapping file to the function of *ml.classify*. The result will be assigned as the attribute *relevance* in each tweet. Given the relevance estimation results, one can render the search result list by retaining those tweets with a value of **1** for the attribute *relevance*.

2) Clustering: Clustering is a type of unsupervised learning method, hence a model file is not needed. However, one needs to specify the configuration parameters depending on the specific algorithm that is applied. For example, the clustering method can be used to cluster the documents according to their themes [32]. This allows us to diversify the microblog content in terms of subtopics:

```
this.theme := ml.cluster(this,[CONFIG],[MAPPING]}
```

The clustering result for each tweet is a cluster tag, which is assigned as an attribute. Therefore, we can diversify the

[15]http://www.cs.waikato.ac.nz/ml/weka/

content conveyed by the tweets by avoiding tweets in the same cluster.

VII. Case Study: Twinder

In this section, we showcase our solution to the problem of building a search engine for Twitter streams with TAP. We start with a prototype system and demonstrate how to accommodate the scientific findings in previous work [9] into the prototype. The goal is to improve the quality of the search results. With the use case of Twinder, we show that our platform allows us to build a search engine for Twitter streams with little effort. For instance, the streaming and the query workflows of the prototype system can be programmed in less than 10 lines of TAL scripts, while the analytics in the improved version can be achieved in less than 40 lines. The implementation of Twinder presented in this paper has been made publicly available[16].

A. Twinder Prototype

We start building a search engine prototype for Twitter streams with two workflows: one streaming and one query workflow. The streaming workflow, which serves for pre-processing the tweets, obtains tweets from Twitter's public stream and adds them into the internal storage and index. The query workflow will be executed whenever a keyword query arrives. A list of tweets will be returned from internal index based on the retrieval score given by the information retrieval approach. Finally, the Twinder prototype will render the results given by the query workflow into a Web page of search results as shown in Figure 2(a).

B. Twinder with Relevance Estimation

In previous work [33], [8], [9], we have found that, besides the retrieval score given by a standard retrieval approach, the semantics extracted from queries as well as the features of tweets can be utilized to improve the retrieval effectiveness. We put these findings into practice and integrate relevance estimation to improve the search result effectiveness – to this end, we expand the two workflows[17]. We apply the tools of semantic enrichment and sentiment analysis in the streaming workflow and add them to the index and storage. The query workflow implements the query expansion approach [33] and prepares the syntactical, semantic, and contextual features that are needed for relevance estimation. Finally, the tweets that are classified as *relevant* will be given to the Web interface for rendering. For demonstration purposes, we marked the *relevant* tweets with *ticks* while the *non-relevant* ones were retained in the list with *crosses* (Figure 2(b)).

[16]https://github.com/ktao/twinder-project

[17]Please refer to https://github.com/ktao/tap/wiki/twinder-example for the TAL scripts of two workflows.

VIII. Conclusions and Future Work

In this paper, we have introduced the **Twitter Analytical Platform**, which enables developers to conduct Twitter data analytics with various analysis tools. The specific analytical tasks can be customized by encoding them in the domain-specific **Twitter Analysis Language** which is also proposed in this paper. We demonstrated the applicability of the TAP by building a prototype search engine for Twitter streams and integrated recently introduced techniques to improve its retrieval effectiveness.

In the future, we plan to further extend the functionality stack of TAP based on the investigation on more use cases and enhance the expressive abilities of TAL, such as loop and conditions. Moreover, we would like to investigate the feasibility of utilizing cloud computing infrastructures to dynamically adapt the processing ability to the demands.

References

[1] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in WWW '10. ACM, 2010, pp. 591–600.

[2] M. Rios and J. Lin, "Distilling Massive Amounts of Data into Simple Visualizations: Twitter Case Studies," in SocMedVis Workshop at ICWSM 2012.

[3] T. Steiner, "Telling Breaking News Stories from Wikipedia with Social Multimedia: A Case Study of the 2014 Winter Olympics." *CoRR*, 2014.

[4] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen, "Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness," in CHI '10. ACM, 2010, pp. 1079–1088.

[5] H. Lietz, C. Wagner, A. Bleier, and M. Strohmaier, "When Politicians Talk: Assessing Online Conversational Practices of Political Parties on Twitter," in ICWSM 2014. The AAAI Press, 2014.

[6] P. T. Metaxas, E. Mustafaraj, and D. Gayo-Avello, "How (Not) to Predict Elections," in PASSAT/SocialCom 2011. IEEE, 2011, pp. 165–171.

[7] Q. Gao, F. Abel, G.-J. Houben, and Y. Yu, "A Comparative Study of Users' Microblogging Behavior on Sina Weibo and Twitter," in UMAP 2012. Springer, 2012, pp. 88–101.

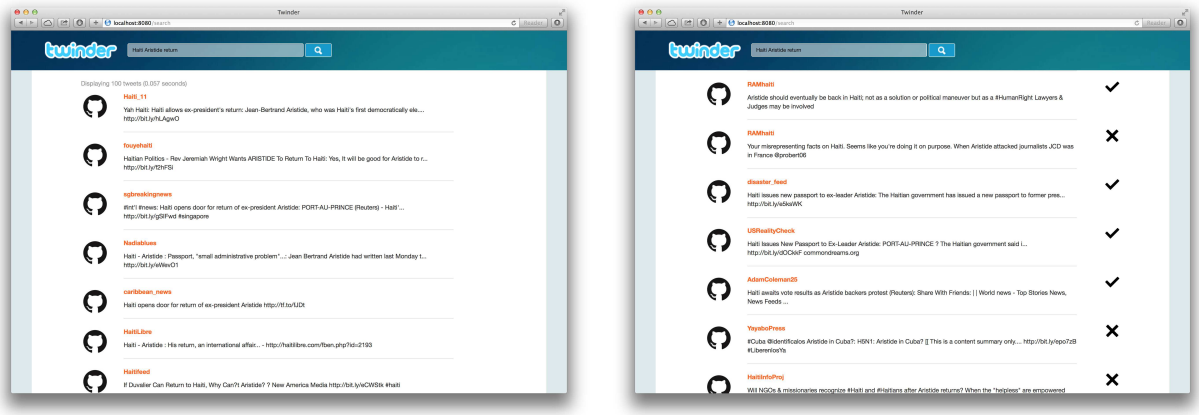
[8] K. Tao, F. Abel, C. Hauff, and G.-J. Houben, "What Makes a Tweet Relevant for a Topic?" in MSM Workshop at WWW 2012. ACM, 2012, pp. 49–56.

[9] K. Tao, F. Abel, C. Hauff, and G.-J. Houben, "Twinder: A Search Engine for Twitter Streams," in ICWE '12. Springer, 2012, pp. 153–168.

[10] K. Tao, F. Abel, C. Hauff, G.-J. Houben, and U. Gadiraju, "Groundhog Day: Near-duplicate Detection on Twitter," in WWW '13. ACM, 2013, pp. 1273–1284.

[11] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in WWW '10.ACM, 2010, pp. 851–860.

[12] P. S. Earle, D. C. Bowden, and M. Guy, "Twitter earthquake detection: earthquake monitoring in a social world," *Annals of Geophysics*, vol. 54, no. 6, 2012.



(a) Twinder Prototype

(b) Twinder with Relevance Estimation

Figure 2. The screenshots of Twinder prototype and the improved version with Relevance Estimation function integrated

[13] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao, "Semantics + Filtering + Search = Twitcident. Exploring Information in Social Web Streams," in HT '12. ACM, 2012, pp. 285–294.

[14] T. Götz and O. Suhre, "Design and Implementation of the UIMA Common Analysis System," *IBM Syst. J.*, vol. 43, no. 3, pp. 476–489, Jul. 2004.

[15] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing User Modeling on Twitter for Personalized News Recommendations," in UMAP 2011. Springer, July 2011, pp. 1–12.

[16] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, "From tweets to polls: Linking text sentiment to public opinion time series," in ICWSM 2010. The AAAI Press, 2010.

[17] A. Stewart, M. Smith, and W. Nejdl, "A Transfer Approach to Detecting Disease Reporting Events in Blog Social Media," in HT '11. ACM, 2011, pp. 271–280.

[18] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.

[19] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web," in ESWC '11, Heraklion, Greece, May 2011.

[20] F. Abel, C. Hauff, G.-J. Houben, and K. Tao, "Leveraging User Modeling on the Social Web with Linked Data," in ICWE '12. Springer, 2012, pp. 378–385.

[21] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, *et al.*, "Building Watson: An overview of the DeepQA project," *AI magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[22] B. Solis and D. Breakenridge, *Putting the Public Back in Public Relations*. Pearson Education, 2009.

[23] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing Temporal Dynamics in Twitter Profiles for Personalized Recommendations in the Social Web," in ACM WebSci '11. ACM, June 2011.

[24] Štefan Dlugolinský, P. Krammer, M. Ciglan, M. Laclavík, and L. Hluchý, "Combining Named Entity Recognition Methods for Concept Extraction in Microposts," in MSM Workshop at WWW '14. pp. 34–41.

[25] Q. Gao, F. Abel, and G.-J. Houben, "GeniUS: Generic User Modeling Library for the Social Semantic Web," in JIST 2011. Springer, December 2011.

[26] K. Tao, F. Abel, Q. Gao, and G.-J. Houben, "TUMS: Twitter-Based User Modeling Service," in ESWC 2011 Workshop. Springer, 2011, pp. 269–283.

[27] L. C. Kats and E. Visser, "The Spoofax Language Workbench: Rules for Declarative Specification of Languages and IDEs," in OOPSLA '10. ACM, 2010, pp. 444–463.

[28] G. D. P. Konat, L. C. L. Kats, G. Wachsmuth, and E. Visser, "Declarative Name Binding and Scope Rules," in SLE '12, Springer, 2012, pp. 311–331.

[29] M. Bravenboer, K. T. Kalleberg, R. Vermaas, and E. Visser, "Stratego/XT 0.17. A language and toolset for program transformation," *Science of Computer Programming*, vol. 72, no. 12, pp. 52 – 70, 2008.

[30] T. Vollebregt, L. C. L. Kats, and E. Visser, "Declarative specification of template-based textual editors," in LDTA '12. ACM, 2012, pp. 8:1–8:7.

[31] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "Exploiting Semantic Annotations for Clustering Geographic Areas and Users in Location-based Social Networks," in 2011 ICWSM Workshop. AAAI, 2011.

[32] C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A Survey of Web Clustering Engines," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 17:1–17:38, Jul. 2009.

[33] K. Tao, F. Abel, and C. Hauff, "WISTUD at TREC 2011: Microblog Track: Exploiting Background Knowledge from DBpedia and News Articles for Search on Twitter," in TREC 2011. NIST, 2011.