Data Enrichment using Data Sources on the Web

Karthik Gomadam, Peter Z. Yeh, Kunal Verma
Accenture Technology Labs
50, W. San Fernando St,
San Jose, CA
{ karthik.gomadam, peter.z.yeh, k.verma} @accenture.com

Abstract

As businesses seek to monetize their data, they are leveraging Web-based delivery mechanisms to provide publicly available data sources. Also, as analytics becomes a central part of many business functions such as customer segmentation, competitive intelligence and fraud detection, many businesses are seeking to enrich their internal data records with data from these data sources. As the number of sources with varying degrees of accuracy and quality proliferate, it is a non-trivial task to effectively select which sources to use for a particular enrichment task. The old model of statically buying data from one or two providers becomes inefficient because of the rapid growth of new forms of useful data such as social media and the lack of dynamism to plug sources in and out. In this paper, we present the data enrichment framework, a tool that uses data mining and other semantic techniques to automatically guide the selection of sources. The enrichment framework also monitors the quality of the data sources and automatically penalizes sources that continue to return low quality results.

Introduction

As enterprises become more data and analytics driven, many businesses are seeking to enrich their internal data records with data from data sources available on the Web. Consider the example, where the consumer database of a company might have the name and address of its consumers. Being able to use publicly available data sources such as LinkedIn, White Pages and Facebook to find information such as employment details and interests, can help the company collect additional features for tasks such as customer segmentation. Currently, this is done in a static fashion where a business buys data from one or two sources and statically integrates with their internal data. However, this approach has the following shortcomings:

1. **Inconsistencies in input data**: It is not uncommon to have different set of missing attributes across records in the input data. For example, for some consumers, the street address information might be missing and for others, information about the city might be missing. To address this issue, one must be able to select the data sources at a granular level, based on the input data that is available and the data sources that can be used.
2. **Quality of a data source may vary, depending on the input data**: Calling a data source with some missing values (even though they may not be mandatory), can result in poor quality results. In such a scenario, one must be able to find the missing values, before calling the source or use an alternative.

In this paper, we present an overview of a data enrichment framework which attempts to automate many sub-tasks of data enrichment. The framework uses a combination of data mining and semantic technologies to automate various tasks such as calculating which attributes are more important than others for source selection, selecting sources based on information available about a data record and past performance of the sources, using multiple sources to reinforce low confidence values, monitoring the utility of sources, as well as adaptation of sources based on past performance. The data enrichment framework makes the following contributions:

1. **Dynamic selection of data sources**: We have developed a novel approach to automatically select the appropriate sequence of data sources to call, based on the available data.
2. **Automated assessment of data source utility**: Our data enrichment algorithm measures the quality of the output of a data source and its overall utility to the enrichment process.
3. **Automated adaptation of data source usage**: Using the data availability and utility scores from prior invocations, the enrichment framework penalized or rewards data sources, which affects how often they are selected.

Motivation

We motivate the need for data enrichment through three real-world examples gathered from large Fortune 500 companies that are clients of Accenture.

Creating comprehensive customer models: Creating comprehensive customer models has become a holy grail in the consumer business, especially in verticals like retail and healthcare. The information that is collected impacts key decisions like inventory management, promotions and rewards, and for delivering a more personal experience. Often businesses engage customers to register for programs like reward cards or connect with the customers using social media. This limited initial engagement gives them the access to basic information about a customer, such as name, email, address, and social media handles. However, in a vast majority of the cases, such information is incomplete and the gaps are not uniform. For example, for a customer John Doe, a business might have the name, street address, and a phone number, whereas for Jane Doe, the available information will be name, email, and a Twitter handle. Leveraging the basic information and completing the gaps, also called as creating a 360 degree customer view has many applications, including:

1. **Personalized and targeted promotions**: The more information a business has about its customer, the better it can personalize deals and promotions. Further, business could also use aggregated information (such as interests of people living in a neighborhood) to manage inventory and run local deals and promotions.
2. **Better segmentation and analytics**: The provider may need more information about their customers, beyond what they have in the profile, for better segmentation and analytics. For example, a certain e-commerce site may know a persons browsing and buying history on their site and have some limited information about the person such as their address and credit card information. However, understanding their professional activities and hobbies may help them get more features for customer segmentation that they can use for suggestions or promotions.
3. **Fraud detection**: The provider may need more information about their customers for detecting fraud. Providers typically create detailed customer profiles to predict their behaviors and detect anomalies. Having demographic and other attributes such as interests and hobbies helps building more accurate customer behavior profiles. Most e-commerce providers are typically under a lot of pressure to detect fraudulent activities on their site as early as possible, so that they can limit their exposure to lawsuits, compliance laws or even loss of reputation.

Current approaches to addressing this challenge largely revolve around subscribing to data sources like Experian. This approach has the following shortcomings:

1. The enrichment task is restricted to the attributes provided by the one or two data sources that they buy from. If they need some other attributes about the customers, it is hard to get them.
2. The selected data sources may have high quality information about attributes, but poor quality about some others. Even if the e-commerce provider knows about other sources, which have those attributes, it is hard to manually integrate more sources.
3. There is no good way to monitor if there is any degradation in the quality of data sources.

Using the enrichment framework in this context would allow the e-commerce provider to dynamically select the best set of sources for a particular attribute in particular data enrichment task. The proposed framework can switch sources across customer records, if the most preferred source does not have information about some attributes for a particular record. For low confidence values, the proposed system uses reconciliation across sources to increase the confidence of the value. The enrichment framework can also continuously monitor and downgrade sources, if there is any loss of quality.

Capital Equipment Maintenance: Companies within the energy and resources industry have significant investments in capital equipments (i.e. drills, oil pumps, etc.). Accurate data about these equipments (e.g. manufacturer, model, etc.) is paramount to operational efficiency, proper maintenance, etc.

The current process for capturing this data begins with manual entry. Followed by manual, periodic “walk-downs” to confirm and validate this information. However, this process is error-prone, and often results in incomplete and inaccurate data about the equipments.

This does not have to be the case. A wealth of structured data sources (e.g. from manufacturers) exist that provides much of the incomplete, missing information. Hence, a solution that can automatically leverage these sources to enrich existing, internal capital equipment data can significantly improve the quality of the data, which in turn can improve operational efficiency and enable proper maintenance.

Competitive Intelligence. The explosive growth of external data (i.e. data outside the business such as Web data, data providers, etc.) can enable businesses to gather rich intelligence about their competitors. For example, companies in the energy and resources industry are very interested in competitive insights such as where a competitor is drilling (or planning to drill); disruptions to drilling due to accidents, weather, etc.; and more.

To gather these insights, companies currently purchase relevant data from third party sources – e.g. IHS and Dodson are just a few examples of third party data sources that aggregate and sell drilling data – to manually enrich existing internal data to generate a comprehensive view of the competitive environment. However, this current process is manual one, which makes it difficult to scale beyond a small handful of data sources. Many useful, data sources that are open (public access) (e.g. sources that provide weather data based on GIS information) are omitted, resulting in gaps in the intelligence gathered.

A solution that can automatically perform this enrichment across a broad range of data sources can provide more in-depth, comprehensive competitive insight.

Overview of Data Enrichment Algorithm

Our Data Enrichment Framework (DEF) takes two inputs – 1) an instance of a data object to be enriched and 2) a set of data sources to use for the enrichment – and outputs an enriched version of the input instance.

DEF enriches the input instance through the following steps. DEF first assesses the importance of each attribute in the input instance. This information is then used by DEF to guide the selection of appropriate data sources to use. Finally, DEF determines the utility of the sources used, so it can adapt its usage of these source (either in a favorable or unfavorable manner) going forward.

Preliminaries

A data object D is a collection of attributes describing a real-world object of interest. We formally define D as $\{a_1, a_2, ...a_n\}$ where a_i is an attribute.

An instance d of a data object D is a partial instantiation of D – i.e. some attributes a_i may not have an instantiated value. We formally define d as having two elements d_k and d_u. d_k consists of attributes whose values are known (i.e. instantiated), which we define formally as $d_k = \{< a, v(a), k_a, k_{v(a)} > ...\}$, where $v(a)$ is the value of attribute a, k_a is the importance of a to the data object D that d is an instance of (ranging from 0.0 to 1.0), and $k_{v(a)}$ is the confidence in the correctness of $v(a)$ (ranging from 0.0 to 1.0). d_u consists of attributes whose values are unknown and hence the targets for enrichment. We define d_u formally as $d_u = \{< a, k_a > ...\}$.

Attribute Importance Assessment

Given an instance d of a data object, DEF first assesses (and sets) the importance k_a of each attribute a to the data object that d is an instance of. DEF uses the importance to guide the subsequent selection of appropriate data sources for enrichment (see next subsection).

Our definition of importance is based on the intuition that an attribute a has high importance to a data object D if its values are highly unique across all instances of D. For example, the attribute *e-mail contact* should have high importance to the *Customer* data object because it satisfies this intuition. However, the attribute *Zip* should have lower importance to the *Customer* object because it does not – i.e. many instances of the *Customer* object have the same zipcode.

DEF captures the above intuition formally with the following equation:

$$k_a = \sqrt{\frac{X^2}{1 + X^2}} \tag{1}$$

where,

$$X = H_{N(D)}(a) \left(\frac{U(a, D)}{|N(D)|} \right) \tag{2}$$

and

$$H_{N(D)}(a) = -\sum_{v \in a} P_v log P_v \tag{3}$$

$U(a, D)$ is the number of unique values of a across all instance of the data object D observed by DEF so far, and $N(D)$ is all instances of D observed by DEF so far. $H_{N(D)}(a)$ is the entropy of the values of a across $N(D)$, and serves as a proxy for the distribution of the values of a.

We note that DEF recomputes k_a as new instances of the data object containing a are observed. Hence, the importance of an attribute to a data object will change over time.

Data Source Selection

DEF selects data sources to enrich attributes of a data object instance d whose values are unknown. DEF will repeat this step until either there are no attributes in d whose values are unknown or there are no more sources to select.

DEF considers two important factors when selecting the next best source to use: 1) whether the source will be able to provide values if called, and 2) whether the source targets unknown attributes in d_u (esp. attributes with high importance). DEF satisfies the first factor by measuring how well known values of d match the inputs required by the source. If there is a good match, then the source is more likely to return values when it's called. DEF also considers the number of times a source was called previously (while enriching d) to prevent "starvation" of other sources.

DEF satisfies the second factor by measuring how many high-importance, unknown attributes the source claims to provide. If a source claims to provide a large number of these attributes, then DEF should select the source over others. This second factor serves as the selection bias.

DEF formally captures these two considerations with the following equation:

$$F_s = \frac{1}{2^{M-1}} B_s \frac{\sum\limits_{a \in d_k \cap I_s} k_{v(a)}}{|I_s|} + \frac{\sum\limits_{a \in d_u \cap O_s} k_a}{|d_u|} \tag{4}$$

where B_s is the base fitness score of a data source s being considered (this value is randomly set between 0.5 and 0.75 when DEF is initialized), I_s is the set of input attributes to the data source, O_s is the set of output attributes from the data source, and M is the number of times the data source has been selected in the context of enriching the current data object instance.

The data source with the highest score F_s that also exceeds a predefined minimum threshold R is selected as the next source to use for enrichment.

For each unknown attribute a' enriched by the selected data source, DEF moves it from d_u to d_k, and computes the confidence $k_{v(a')}$ in the value provided for a' by the selected source. This confidence is used in subsequent iterations of the enrichment process, and is computed using the following formula:

$$k_{v(a')} = \begin{cases} e^{\left(\frac{1}{|V_{a'}|} - 1\right)} W & \text{, if } k_{v(a')} = Null \\ e^{\lambda (k_{v(a')} - 1)} & \text{, if } k_{v(a')} \neq Null \end{cases} \tag{5}$$

where,

$$W = \frac{\sum\limits_{a \in d_k \cap I_s} k_{v(a)}}{|I_s|} \tag{6}$$

W is the confidence over all input attributes to the source, and $V_{a'}$ is the set of output values returned by a data source for an unknown attribute a'.

This formula captures two important factors. First, if multiple values are returned, then there is ambiguity and hence the confidence in the output should be discounted. Second, if an output value is corroborated by output values given by previously selected data sources, then the confidence should be further increased. The λ factor is the corroboration factor (< 1.0), and defaults to 1.0.

In addition to selecting appropriate data sources to use, DEF must also resolve ambiguities that occur during the enrichment process. For example, given the following instance of the *Customer* data object:

(Name: John Smith, City: San Jose, Occupation: NULL)

a data source may return multiple values for the unknown attribute of *Occupation* (e.g. Programmer, Artist, etc).

To resolve this ambiguity, DEF will branch the original instance – one branch for each returned value – and each branched instance will be subsequently enriched using the same steps above. Hence, a single data object instance may result in multiple instances at the end of the enrichment process.

DEF will repeat the above process until either d_u is empty or there are no sources whose score F_s exceeds R. Once this process terminates, DEF computes the fitness for each resulting instance using the following equation:

$$\frac{\sum\limits_{a \in d_k \cap d_U} k_{v(a)} k_a}{|d_k \cup d_u|} \tag{7}$$

and returns top K instances.

Data Source Utility Adaptation

Once a data source has been called, DEF determines the utility of the source in enriching the data object instance of interest. Intuitively, DEF models the utility of a data source as a "contract" – i.e. if DEF provides a data source with high confidence input values, then it is reasonable to expect the data source to provide values for all the output attributes that it claims to target. Moreover, these values should not be generic and should have low ambiguity. If these expectations are violated, then the data source should be penalized heavily.

On the other hand, if DEF did not provide a data source with good inputs, then the source should be penalized minimally (if at all) if it fails to provide any useful outputs.

Alternatively, if a data source is able to provide unambiguous values for unknown attributes in the data object instance (esp. high importance attributes), then DEF should reward the source and give it more preference going forward.

DEF captures this notion formally with the following equation:

$$U_s = W\left(\frac{1}{|O_s|}\left(\sum_{a \in O_s^+} e^{\frac{1}{|V_a|}-1} k_a^{P_T v(a)} - \sum_{a \in O_s^-} k_a\right)\right) \tag{8}$$

where,

$$P_T(v(a)) \quad = \quad \begin{cases} P_T(v(a)) & \text{, if } |V_a| = 1 \\ \operatorname*{argmin}\limits_{v(a) \in V_a} P_T(v(a)) & \text{, if } |V_a| > 1 \end{cases} \tag{9}$$

O_s^+ are the output attributes from a data source for which values were returned, O_s^- are the output attributes from the same source for which values were not returned, and $P_T(v(a))$ is the relative frequency of the value $v(a)$ over the past T values returned by the data source for the attribute a. W is the confidence over all input attributes to the source, and is defined in the previous subsection.

The utility of a data source U_s from the past n calls are then used to adjust the base fitness score of the data source. This adjustment is captured with the following equation

$$B_s = B_s + \gamma \frac{1}{n} \sum_{1}^{n} U_s(T-i) \tag{10}$$

where B_s is the base fitness score of a data source s, $U_s(T-i)$ is the utility of the data source i time steps back, and γ is the adjustment rate.

System Architecture

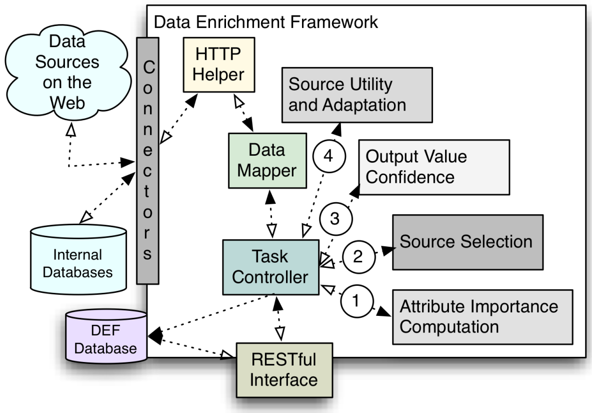

Figure 1: Design overview of enrichment framework

The main components of DEF are illustrated in Figure 1. The task manager starts a new enrichment project by instantiates and executes the enrichment engine. The enrichment engine uses the attribute computation module to calculate the attribute relevance. The relevance scores are used in source selection. Using the HTTP helper module, the engine then invokes the connector for the selected data source. A connector is a proxy that communicates with the actual data source and is a RESTful Web service in itself. The enrichment framework requires every data source to have a connector and that each connector have two operations: 1) a return_schema GET operation that returns the input and output schema, and 2) a get_data POST operation that takes as input the input for the data source as POST parameters and returns the response as a JSON. For internal databases, we have special connectors that wrap queries as RESTful end points. Once a response is obtained from the connector, the enrichment engine computes the output value confidence, applies the necessary mapping rules, and integrates the response with the existing data object. In addition to this, the source utility is also computed. The mapping, invocation, confidence and source utility value computation steps are repeated until either all values for all attributes are computed or if all sources have been invoked. The result is then written into the enrichment database.

In designing the enrichment framework, we have adopted a service oriented approach, with the goal of exposing the enrichment framework as a "platform as a service". The core tasks in the framework are exposed as RESTful end points. These include end points for creating data objects, importing datasets, adding data sources, and for starting an enrichment task. When the "start enrichment task" task resource is invoked and a task is successfully started, the framework responds with a JSON that has the enrichment task identifier. This identifier can then be used to GET the enriched data from the database. The framework supports both batch GET and streaming GET using the comet (Mahemoff 2006) pattern.

Data mapping is one of the key challenges in any data integration system. While extensive research literature exists for automated and semi-automated approaches for mapping and matching (Madhavan et al. 2003; Doan, Domingos, and Halevy 2001) it is our observation that these techniques do not guarantee the high-level of accuracy required in enterprise solutions. So, we currently adopt a manual approach, aided by a graphical interface for data mapping. The source and the target schemas are shown to the users as two trees, one to the left and one to the right. Users can select the attributes from the source schema and draw a line between them and attributes of the target schema. Currently, our mapping system supports assignment, merge, split, numerical operations, and unit conversions. When the user saves the mappings, the maps are stored as mapping rules. Each mapping rule is represented as a tuple containing the source attributes, target attributes, mapping operations, and conditions. Conditions include merge and split delimiters and conversion factors.

Challenges and Experiences

Data sources (exposed as RESTful services and/or Web APIs) are an integral part of the data enrichment framework. Even though data sources have been available for many years, their traction within the enterprise has been minimal. During the development of the enrichment framework, we got a few insights into the reasons for this lack of adoption. Rate limiting (especially in the number of calls that can be made within a short interval of time) and lack of syndication processes, make it hard to reliably use a data source, especially in client deployments. Further, many of the data sources do not offer explicit SLA driven contracts, thus making them unattractive. Poor API documentation is another reason. Often times, when a developer is using an API, they find that some of the capabilities that are present in the UI driven version of the service (such as LinkedIn API vs LinkedIn.com) are either absent or are not sufficiently documented. This makes it harder to understand the actual capabilities of an API. Further, the data formats for the input and output are often described in text (as opposed to having a JSON / XML snippet), adding to the complexity. API providers do not "push" API changes to the developer. In most cases, developers find out about the change after a failed call to the API. We feel that the above mentioned reasons play a significant role in impeding the adoption of data sources APIs in enterprise software development.

Related Work

Mashups or Web application hybrids (Merrill 2006) have emerged as the popular approach for integrating data from different data sources on the Web. Plethora of tools such as (Maximilien et al. 2007; Fagan 2007; Software") have been developed for creating mashups. However, in a traditional mashup, the data sources used and the order in which they are invoked are static. In the context of data enrichment, such a static approach may not yield desired results, since the gaps in a dataset are often not homogeneous. The granular selection and ordering of sources that we advocate in this paper is the main difference between our work and the standard mashup techniques. An evaluation of approaches for data mediation in mashup tools is presented in (Di Lorenzo et al. 2009). While this paper compares different approaches, it does not in itself advocate a novel method. (Maximilien et al. 2007) present a domain specific language for developing mashups. The paper proposes a manual mediation approach, similar to the approach discussed in this work. However, the DSL driven platform does not support run-time source selection based on available attributes and does not support source adaptation.

Conclusion

In this paper we present a framework for data enrichment using data sources on the Web. The salient features of our system include the ability to dynamically select the appropriate sequence of data sources to use, based on the available data. We also discuss approaches for automatically computing the utility of data sources and adapting to their usage. The framework is exposed (internally) as a "platform as a service", accessible via RESTful APIs. We are currently in the process of piloting the system in the energy and resources domain.

References

Di Lorenzo, G.; Hacid, H.; Paik, H.; and Benatallah, B. 2009. Data integration in mashups. *ACM Sigmod Record* 38(1):59–66.

Doan, A.; Domingos, P.; and Halevy, A. 2001. Reconciling schemas of disparate data sources: A machine-learning approach. In *ACM Sigmod Record*, volume 30, 509–520. ACM.

Fagan, J. 2007. Mashing up multiple web feeds using yahoo! pipes. *Computers in Libraries* 27(10):8.

Madhavan, J.; Bernstein, P. A.; Chen, K.; Halevy, A. Y.; and Shenoy, P. 2003. Corpus-based schema matching. In *IIWeb*, 59–63.

Mahemoff, M. 2006. Design patterns for ajax. `http://ajaxpatterns.org/HTTP_Streaming`.

Maximilien, E.; Wilkinson, H.; Desai, N.; and Tai, S. 2007. A domain-specific language for web apis and services mashups. *Service-oriented computing–ICSOC 2007* 13–26.

Merrill, D. 2006. Mashups: The new breed of web app. *IBM Web Architecture Technical Library* 1–13.

Software", K. Open kapow. `http://kapowsoftware.com/`.