

Self-Service Data Discovery and Information Sharing

Fostering the Engineering Capacity of the Data-Driven Mindset

Kurt Englmeier¹ and Hugo Román²

¹Faculty of Computer Science, Schmalkalden University of Applied Science, Blechhammer, Schmalkalden, Germany

²Soluciones S.A., 33 Nueva York, Santiago, Chile

Keywords: Data Discovery, Information Integration, Language Pattern, Natural Language, Data Governance, Simplicity, Information Extraction, Metadata.

Abstract: People dealing with information in IT-based environments tend to develop a data-driven mindset that constitutes shallow engineering knowledge and turns them into tech-savvy information consumers. We argue that these consumers can manage data discovery and information sharing on their own without explicit support from IT. We argue furthermore that user-driven discovery can even be mandatory when mainstream discovery concentrates on facts appearing massively in data and bypasses the little and unspectacular facts consumers expect to discover in their data. Finally we argue that tech-savvy users can depict blueprints of these facts using a pattern language that combines the user's work jargon with a simple syntax. We present a working solution for such an easy-to-learn pattern language for self-service data discovery and information sharing (DISL). The language has been developed in an industry-academia partnership and was applied in the area of assessment in the real estate sector in Chile. A prototypical discovery service operating on DISL gathers information from contracts and related certificates and prepares the discovered facts for information sharing.

1 INTRODUCTION

Information discovery requires, in general, profound technical and linguistic skills. Systems designed in close cooperation between IT specialists and knowledge management experts serve the need for mainstream discovery from business and other areas. Mainstream discovery addresses facts that are massively observable in data and thus commonly described in global metadata descriptions like schema.org or addressed by the organization's master data management. Example addressees are named entities like "organization", or "person", "location", monetary expressions, or temporal expressions. Pattern recognition as well as linguistic and probabilistic methods serve to identify candidates in data that match these schemas. Mainstream discovery defines also the qualities that make up the facts to be discovered. These definitions are not quite sensitive to the peculiarities of individual discovery requests. They have to serve data patterns that are generic enough in order to justify the development of the corresponding discovery systems from an economic point of view. Industrial practice,

however, shows that there are many "light-weight" discovery requests that are not addressed by mainstream discovery. The inclusion of the information consumers' individual requirements into application development is usually considered as too expensive or too time-consuming. Further-more, their requests tend to change dynamically. Even when addressing the same data collection, different users are likely interested in different facts and, furthermore, this interest may vary over time. Users may analyze complaints of clients about machine failures, for instance. One user may be interested in the affected part of the machine, while the other one wants to explore the cause of the failure. In many a case, discovery and its corresponding data descriptions even serve only a one-time purpose, that is, they are disposable artifacts. In short, discovery can address very specific, small-scale, and ad hoc contexts that lie outside the scope of mainstream information discovery. Consequently, users have to handle many of their individual requests manually, they have to check data and extract the required facts by hand.

We present here a working solution for self

-service IT that enables tech-savvy end-users to design and manage their discovery processes and to share their findings with their colleagues. Its objective is merging smoothly two knowledge areas, namely domain knowledge and engineering knowledge required for “light-weight” data extraction and information sharing. We want to encourage users to translate their vision of the facts they expect to discover in data directly into machine-processable instructions.

In this article we explain the principles of self-service information discovery and the role and significance of the information consumer’s data-driven mindset. The practical part outlines our data discovery and information sharing language (DISL), emerging from the rationale of self-service discovery. It also provides results from the first application in the area of assessment in the real estate sector and describes how data integration and information sharing is supported. Even though DISL focuses on discovery in unstructured data in general, here we concentrate on its application on texts.

2 DATA-DRIVEN MINDSET

What we want to discover in data is facts. For instance, we want to know “the birthdate of Bert Brecht” or the “the age of the broken machine part, as indicated by the customer in his mail”. The facts emerge from data that depict certain qualities. With a blueprint of these facts in mind we can steer our discovery process and detect these facts in structured as well as unstructured data.

People working with information have a data-driven mindset per se (Pentland, 2013; Viaene, 2013), that is, they resort to mental models (Norman, 1987) that abstractly reflect the facts they expect to encounter in their information environment (Brandt and Uden, 2003). This mindset enables them to sketch blueprints of the things they are looking for. We argue that their computer literacy helps them to express these blueprints in forms that can be processed by machines. These expressions are far from being programming instructions but reflect the users’ “natural” engineering knowledge. The machine then takes the blueprints and identifies these facts in data, even though the blueprint abstracts away many details of the facts.

Experimenting with data is an essential attitude that stimulates data discovery experiences. People initiate and control discovery by a certain belief - predisposition or bias reinforced by years of expertise - and gradually refine this belief. They

gather data, try their hypotheses in a sandbox first and check the results against their blueprints, and then, after sufficient iterations, they operationalize their findings in their individual world and then discuss them with their colleagues. After having thoroughly tested their hypotheses, information consumers institutionalize them to their corporate world, that is, cultivate them in their information ecosystem. The language knowledge and their mental models constitute shallow knowledge necessary and sufficient to engineer statements that are processable by the discovery services (Sawyer et al., 2005).

The blueprints thus serve two purposes: they reflect semantic qualities of the facts the discovery engine shall locate and extract. Simultaneously, they are the building blocks of the meta-language that, when correctly syndicated, support data integration and sharing. While syndicating metadata along their domain competence, users foster implicitly active compliance with organizational data governance policies.

The design of DISL is guided by the paradigm of simplicity in IT (Magaria and Hinchey, 2013). We want users to concentrate on discovery and sharing, to stick to their data-driven mindset, and to express their requests as far as possible in their own language. A good starting point therefore is the workplace jargon of information consumers. To avoid any irritations or ambiguities, people try to be quite precise in their descriptions. Even though these descriptions are composed of natural language terms and statements, humans are quite good in safeguarding literal meaning in their descriptions (Iwanska, 2000). We avail ourselves of literal meaning because we can interpret statements correctly in the absence of any further explicit and implicit context. This aspect is also important when it later comes to machine interpretation of these statements. In the end, we need discovery engines that support semantic search (Ding et al., 2005; Zhao, 2007). They operate on data enriched by annotations that make implicit meaning explicit. They also set up the semantic skeleton of the information ecosystem where search and discovery happens. This skeleton represented as a network of metadata emerges from the domain knowledge of the information consumers. For us, the data-driven mindset becomes evident when users can turn their domain and shallow engineering knowledge into machine instructions suitable for the precise detection and extraction of the facts they expect to discover.

3 SELF-SERVICE DISCOVERY

Information discovery starts with information extraction (IE) (Cowie and Lehnert, 1996) that distils text or even scattered documents to a germ of the original raw material. IT experts engineer information extraction systems that operate on templates for the facts to be extracted. Labelled slots constitute these templates whereby the labels represent annotated terms.

Self-service information discovery starts with user-driven IE. The users first have to engineer their extraction templates that can also be considered as entity recognizers. This means, a certain amount of engineering is indispensable in IE. The key question is whether information consumers have the necessary engineering knowledge to handle discovery services on their own. This shallow knowledge is assumed to be acquired easily and thoroughly specific to the task at hand (Fan et al., 2012). The assumption in self-service discovery is that users with their domain competence and shallow engineering knowledge are in the position to manage a certain level of data discovery and information sharing on their own.

The users' blueprints may be simple and concise, but they are comprehensive enough to cover their request. This, in turn, fosters the control of the discovery process. A template with, say eight to twelve slots, can comprehensively cover real-world requests in small-scale domains. This low level of complexity makes it easy for the information consumer to manually control discovery. Whenever they encounter unfilled slots or mistakenly filled slots they may check the corresponding document for obvious errors. On the other hand, they may adapt their blueprint if the representation of a fact appears to be sound in text, but the slots do not consistently correspond to the qualities of the fact.

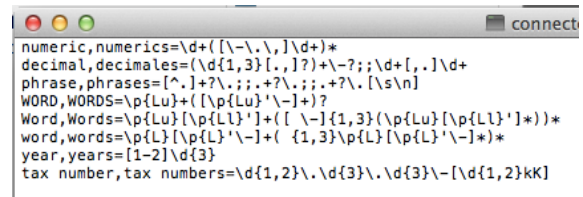
3.1 Basic Language Elements

IE knows a lot of methods to match text excerpts (i.e. candidates for information entities) with slot keys (i.e. labels) and to fill the slots. In our approach we focus on pattern recognition. Each template slot is thus linked to a descriptive pattern that is rendered as Regular Expression in combination with key terms. Regular Expressions are a powerful instrument to precisely detect all kind of patterns in data. Their application is in particular useful for the detection of facts in unstructured information. Furthermore, they offer the opportunity to sketch a variety of patterns for a particular fact that may appear in many variant forms. A date, for instance,

can be expressed in different forms even within the same language. Regular Expressions have a particular advantage, too, if data represent a named entity, e.g. as combination of numerical data and words. A birthdate may be rendered by keywords (“born on”) or symbols (an asterisk) forming a specific pattern with an adjacent date. The position within text may also suffice to qualify a date as birthday: “Franz Kafka (3 July 1883 – 3 June 1924)”.

In each text we can identify numerous facts of different complexities. Many of them can be identified in patterns that comprise facts located in close proximity. Information on a person may span over more basic facts like name, tax payer number, birthdate, address, etc. Some fact may comprise constituent facts that are widely scattered over a broader area of the data, even over more text pages or documents for instance.

Regular Expressions are a powerful, but absolutely not a user-friendly instrument. They require special skills and are not easy to handle, in particular, when they are addressing complex, i.e. real-world, patterns. Besides, Regular Expressions representing high level facts are extremely complex and barely manageable, even for professionals. Their application also has limitations, when relevant elements (qualities) of facts are too dispersed over the data set, that means when too much “noise” appears between facts or their qualities. We therefore propose a language for discovery and sharing that adopts Regular Expressions but shields users from their complexity. IT provides users with a stock of labelled Regular Expressions addressing entities like “word”, “tax payer number”, “phrase”, “decimal” etc. (see Figure 1). Instead of defining Regular Expressions, the users compose their patterns by resorting to these basic patterns or to the patterns they have already defined by themselves. The DISL syntax serves to describe how facts, as the constituent parts of the fact requested by the user, appear as sequences of word patterns in data. The corresponding descriptive pattern gradually aggregates into the complex pattern for the requested fact.



```
numeric,numerics=\d+{[\-\.\\,]\d+}*
decimal,decimales=(\d{1,3}[.],)?\d+; \d+[\-\.\\,]\d+
phrase,phrases=[^]+?[\-\.\\,];.+?[\-\.\\,]{\s\n}
WORD,WORDS=\p{Lu}+(\p{Lu}'\-|+)?
Word,Words=\p{Lu}[\p{Ll}']+( [\-] {1,3}(\p{Lu}[\p{Ll}'])*)*
word,words=\p{L}[\p{L}'\-]+( {1,3}\p{L}[\p{L}'\-])*
year,years=[1-2]\d{3}
tax number,tax numbers=\d{1,2}\. \d{3}\. \d{3}\-[\d{1,2}kk]
```

Figure 1: Examples of basic patterns (entity recognizer). Usually, IT experts provide these patterns to information consumers that construct their patterns from here.

3.2 The Syntax of DISL

The users achieve the definition of complex descriptive patterns by iteratively aggregating their pattern definitions starting from the basic patterns (see Figure 1) provided by their IT colleagues. This means, any pattern (if not a basic pattern) can be decomposed into smaller units from top to bottom, eventually into basic patterns and key terms. The discovery service sees any pattern as more or less big Regular Expression. We call the patterns at the bottom layer also “connector patterns” because they link to the respective data representation technique, used by the service, in our case Regular Expressions. They act as primitive data types.

Table 1: Operators of the discovery language.

.	The dot indicates strong sequence (“followed by”). The fact indicated before the dot must be located before the one indicated after the dot.
,	Comma means weak sequence. Some of indicated facts (at least one) shall appear sequentially in the data. However, they may appear in any order (inclusive combination).
;	The semicolon is used to indicate an exclusive combination. Just one of the facts ought to be located.
:	Labeling operator: the name after the colon is assigned to facts or a group of facts. Labeling serves the implicit introduction of (local) nested patterns.
(...)	Parentheses serve to indicate a group of facts. Grouping only makes sense together with the labeling operator.
?	The question mark indicates that a fact or group of facts can be optional, that is, the corresponding fact can but need not be located in the data sources.
“keyword”	The discovery service treats keywords indicated between quotation marks as fixed expressions. If it represents a verb, the keyword is reduced to its principal part. Each noun or adjective is expanded to a Regular Expression covering their possible grammatical forms (flections). Three dots (...) within a constant indicate that there may appear a small number of irrelevant terms between the elements of the fixed expressions.

The syntax for the definition of patterns is quite simple. It supports a handful of operators the users

employ to define a descriptive pattern as a sequence of constituent elements, i.e. word patterns.

The following generic definition of a descriptive pattern summarizes the syntax of our DISL for discovery and integration, Table 1 explains in more detail the functionality of the operators:

```
concept, concepts = element1.element2;
element3.(element4,element5):name.?elem
ent6
```

On the left side of the equation the user defines the name of the pattern. The right side of the equation lists the sequence of pattern elements. The pattern can be assigned to a singular term and optionally also to its corresponding plural term. When defining their own patterns, people intuitively apply both forms.

4 EXPERIMENTS AND RESULTS

4.1 Illustrative Example

To illustrate DISL we delegated a request to our discovery engine and let it operate on the (English) Wikipedia collection. We wanted to extract birth and death dates of German writers and to list their works with their original title and the corresponding English translation of the title. By checking a sample

Heiner Müller (January 9, 1929 – December 30, 1995) was a German Described as "the theatre's greatest living poet" since **Samuel Beckett**, after **Bertolt Brecht**. His "enigmatic, fragmentary pieces" are a significant however, with his drama *Die Umsiedlerin* (*The Resettler Woman*) from the Writers' Association in the same year. The East German niere of *Der Bau* (*Construction Site*) in 1965 and censoring his *Mi*

Franz Kafka^[a] (3 July 1883 – 3 June 1924) was a German-language influential authors of the 20th century. Kafka strongly influenced genre *Metamorphosis*"), *Der Prozess* (*The Trial*), and *Das Schloss* (*The Ca*. *Betrachtung* (*Contemplation*) and *Ein Landarzt* (*A Country Doctor* ed the story collection *Ein Hungerkünstler* (*A Hunger Artist*) for ng his novels *Der Prozess*, *Das Schloss* and *Amerika* (also known

Bertolt Brecht (/brekt/^[1]/ˈrekt/^[2]/ˈbreçt/^[3] German: [ˈbɛʁtɔlt ˈbʁɛçt] ^(ⓘ) listen); born ◄ **Eugen Berthold Friedrich Brecht** (help·info); 10 February 1898 – 14 was a German poet, playwright, theatre director, and **Marxist**.

- *The Elephant Calf* (*Das Elefantenkalb*) 1924–26/1926
- *Little Mahagonny* (*Mahagonny-Songspiel*) 1927/1927
- *The Threepenny Opera* (*Die Dreigroschenoper*) 1928/1928

Figure 2: Original text sections from a larger sample of texts. The snippets contain the requested information on the authors' birth and death date and on the titles of their works.

of Wikipedia pages, we familiarized ourselves with the way how these facts are represented (Figure 2) and defined the corresponding descriptive patterns using DSL (Figure 3). The discovery engine applied our blueprints on this collection and returned the extracted information in XML format (Figure 4).

The example illustrates also the rationale for data integration and information sharing behind DSL that supports the collaborative development of a metadata schema. This schema can constitute the semantic skeleton of an information ecosystem on group or organizational level. In this context, self-service discovery and integration supports “active compliance”, that is, the collaborative agreement on a unified overarching metadata schema.

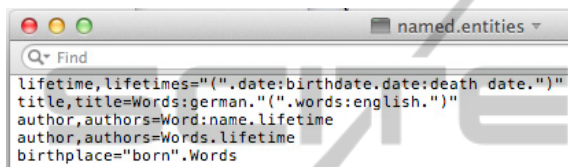


Figure 3: Patterns to discover information on birth and death dates of (German) writers and the titles of their works (original title and English translation). On the left side appear the labels of the patterns.



Figure 4: Example of the results rendered in XML. The discovery engine takes the patterns as defined by the users (similar to those in Figure 3), detects the corresponding data in the texts (like the ones shown in Figure 2), extracts these data and stores them in XML. The tags of the XML elements (or slots) correspond to the pattern labels.

As already said, self-service discovery addresses, among other things, ad hoc requests for discovery that, like in the example above, are not too complex. The integration in terms of consolidating metadata schema on a broader organizational level is thus not the only issue in integration. Many blueprints for small-scale and ad hoc requests are disposable artifacts for individual purposes. The integration into the technical environment is more important for discovery requests that need to be shared. What kind of input formats are supported by the discovery service, how are the results presented? For the time being, our service accepts input such as HTML, PDF, or plain text documents. The output is simply rendered in XML, in order to enable a smooth integration into follow-up processes for data analytics, visualization, reporting and the like. The slots (with annotated terms) capture the facts according to the users’ pattern definitions render them by XML elements. Basic patterns are treated like primitive data types; entity elements that correspond to them are not explicitly tagged.

4.2 Results from the First Application

The design of DSL had a highly experimental character. The examples above relate to experiments we ran with Wikipedia articles on writers. However, we applied DSL and our discovery service on legal texts addressing real estate contracts and related certificates from land registries. The objective was a cross-collection search for comprehensive information on vendors, purchasers, and real estates. The application passed through a number of cycles until it reached a sufficient level of maturity and acceptance. At the same time, the language has been tested by a mixed group of domain experts (lawyers and their assistants), students, and programmers.

The over-arching objective was and still is to ensure that the language has a high level of usability, in particular in terms of learnability, memorability, and ease of use. During the iterative design we got familiar with the shared mental model information consumers have about appearance and nature of descriptive (word) patterns: entities corresponding to an information blueprint can be dispersed over a number of text blocks (facts), for instance: [vendor] sells [object] to [buyer]. Within each block, terms may appear in arbitrary order, like the characteristics of a person or organization. Some terms are optional and there are keywords like “sell” or “buy” that essentially characterize the fact “transaction purchase”. The question then was, how the appearance of terms and term blocks can be expressed by

operators.

The operators as listed above (see Table 1) and their functionality turned out to be “intuitive”, i.e. matching the information consumers’ mental model. There were syntax elements considered as highly intuitive, like the quotation marks indicating some sort of fixed text in an otherwise parametric presentation of a pattern. The users immediately perceived dots as separators of building blocks of their patterns. The role of the comma was apparent, too. However, there were also things to learn, the difference between comma and semicolon, for instance. Some had to learn that the question mark has to be put ahead of the expression to mark it as optional rather than thereafter. For others, however, it was more intuitive for this purpose to have a leading question mark than a trailing one.

We ran our experiments with about 2000 documents (real estate contracts with related certificates) distributed over 18 data sources. In the first place, this sample may seem small to validate our approach or to underpin its scalability. However, the inherent character of basically unstructured data distributed over different sources reflects the nature of the challenge we face in data discovery, even within the context of Big Data. The language applied in contracts and related certificates is quite uniform and not narratively complex. We are convinced that our document sample covers this language in its entirety, and thus scales for even larger collections. In many information ecosystems we barely have to deal with highly complex narrative forms. Due to this fact, we consider our approach as scalable also towards thematic areas outside legal information as long as the narrative nature is relatively uniform, such as is the case for legal texts.

5 CONCLUSION

Our work-in-progress demonstrates the feasibility of self-service data discovery and information sharing. With a simple instrument like DISL the information consumers can leverage their shallow engineering knowledge for managing discovery services on their own. Over the time, information consumers naturally develop a data-driven mindset and with their computer literacy a certain level of “natural” engineering knowledge that enables them to handle these discovery tools, including the tools’ command language. Our experiments indicate that users can develop shallow engineering knowledge without much effort. If the discovery tool requires not more than that level of knowledge they can

transform their domain knowledge easily into machine instructions. This smooth integration of domain and tool knowledge completes the picture of self-service discovery that meanwhile is also demanded by the industry (Sallam et al., 2014).

There are many discovery tasks that serve individual, ad hoc, and transient purposes. Main stream discovery, in contrast, supports reoccurring discovery requests commonly shared by large user communities and operates on large data collection, including sometimes the entire Web. We can conceive manifold scenarios for non-mainstream discovery. Users may have to analyse from time to time dozens of failure descriptions or complaints, for instance. The corresponding data collections are personal or shared among small groups and consist of bunches of PDF files or emails, for instance, barely documents on the Web. Dynamically changing small-scale requests would mean permanent system adaptation, which is too intricate and too expensive in the majority of cases. With a flexible self-service solution like DISL information consumers can reap the benefits of automatic information discovery and sharing and avoid the drawbacks of mainstream discovery.

REFERENCES

- Brandt, D. S., Uden, L., 2003. Insight Into Mental Models of Novice Internet Searchers, *Communications of the ACM*, vol. 46 no. 7, pp. 133-136.
- Cowie, J., Lehnert, W., 1996. Information Extraction, *Communications of the ACM*, vol. 39 no. 1, pp. 80-91.
- Ding, L., Finin, T., Joshi, A., Pan, R., Peng, Y., Reddivari, P., 2005. Search on the Semantic Web, *IEEE Computer*, vol. 38, no. 10, 2005, pp. 62-69.
- Fan, J., Kalyanpur, A., Gondek, D.C., Ferrucci, D.A., 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, vol. 56, no 3.4, pp.: 5:1-5:10
- Iwanska, L.M., 2000. Natural Language Is a Powerful Knowledge Representation System: The UNO Model, in: L.M. Iwanska and S.C. Shapiro (eds.), *Natural Language Processing and Knowledge Representation*, AAAI Press, Menlo Park, USA, pp. 7-64.
- Magaria, T., Hinchey, M., 2013. Simplicity in IT: The Power of Less, *IEEE Computer*, vol. 46, no. 11, pp. 23-25.
- Norman, D., 1987. Some observations on mental models. D. Gentner; A. Stevens, (Eds.) *Mental Models*, Lawrence Erlbaum, Hillsdale, NJ.
- Pentland, A., 2013. The data-driven society. *Scientific American*, vol. 309, no. 4, pp. 64-69.
- Sallam, R., Tapadinhas, J., Parenteau, J., Yuen, D., Hostmann, B., 2014. Magic Quadrant for Business Intelligence and Analytics Platforms, February 2014,

retrieved at: <http://www.gartner.com/technology/reprints.do?id=1-1QYL23J&ct=140220&st=sb> on June 12, 2014.

- Sawyer, P., Rayson, P., Cosh, K., 2005. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *IEEE Transactions on Software Engineering*, vol. 31, no. 11, pp. 969 – 981.
- Viaene, S., 2013. Data Scientists Aren't Domain Experts, *IT Professional*, vol. 15, no. 6, pp. 12-17.
- Zhao, H., 2007. Semantic Matching, *Communications of the ACM*, vol. 50 no. 1, 2007, pp. 45-50.

