Automated Implicit Linking of Open Government Data

by

Johanna E. Flores

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Approved:

_____

James Hendler, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

May, 2011

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This thesis is the culmination of a year's long work that would not have been possible without the patience, guidance and support of many individuals within the Tetherless World Constellation research group and RPI community.

First, my utmost gratitude goes to Dr. James Hendler, Leading Chair of the Tetherless World Constellation, for allowing me to join his research group, which was instrumental in the inception of this work, and for advising my thesis as it progressed. His encouragement and insight will forever be invaluable and his presence is always an inspiration to hold myself to a higher standard.

Li Ding, Research Associate with Tetherless World Constellation, who patiently mentored me as my research was just beginning to take form, and whose sincere interest in my work and progress was always to my benefit.

Timothy Lebo and Dominic DiFranzo, TWC researchers, who, despite having busy schedules of their own, were gracious enough to take the time to give me feedback during the development of my work.

Finally, my undergraduate adviser Professor Szymanski Boleslaw, for his guidance and help during my education at RPI, without which I would have not chosen to follow this path.

ABSTRACT

The proposed approach of automated implicit data linking aims at identifying string literals and implicitly promoting them to named semantic entities. This paper focuses on the specific problem of linking literals within Data.gov datasets, which commonly have no external or crosslinks, to U.S. state and territory URIs, maintained by DBpedia. The approach to identification relies on well-known natural language processing techniques to perform accurate entity resolution, which is necessary for ensuring links between string literals and URIs are correct. Determining whether predicates, attached to the resolved literals, can be placed in the "U.S. state" category is also a key point of interest and vital for verification. Three main heuristics are used to perform verification on predicates and calculate a final certainty value which measures how sure the system is that they are state-related. This paper describes the approach to automated implicit linking in detail and presents an evaluation (precision and recall) of the results returned by the deterministic heuristics on a collection of government datasets.

1. INTRODUCTION

Over the past decade, there has been substantial growth in the amount of data, (both structured and unstructured), that has been published to the internet, by professional and casual data gatherers alike. Advancements in open and comprehensible tools, such as wikis and blogs, have provided a fairly seamless medium for people of varying proficiencies and interests to contribute textual information to the Web. Intuitive formats and specifications for structured data, such as XML and JSON, also allow data to be organized, presented and exchanged easily with little cost via RESTful web services. These technologies have been commonly adopted within social and commercial domains, and are responsible for a steady influx of new information added to the web daily, broadening the infrastructure of the collective human knowledge already present. Relatively recently, under initiative from President Barack Obama, the U.S. government has made an effort to continuously release thousands of its government datasets to the public domain. The government has constructed its own platform, known as Data.gov [1], to publish its data and promote demonstrations and awareness of how the data is being used in real-world problem-solving and applications. The government's datasets are contributed by numerous agencies within the federal government and contain data which vary in granularity and specificity across multiple topics and categories. Of greater interest is the fact that many of these datasets are available in convenient formats such as XML, CSV and RDF, which make the raw data much more enticing and usable to developers and data consumers wishing to incorporate open government data into web applications and services.

As data-related web technologies have continued to progress, an interest in incorporating data semantics and provenance has driven part of this evolution, giving rise to the concept of the Semantic Web. The human-web, (the internet as it is perceived by people), is a network of human-readable (i.e., text, images) webpages interconnected by hyperlinks. The Semantic Web hopes to enhance the content already present by promoting standards and frameworks to include machine-consumable metadata within human-readable pages and other data sources, both structured and unstructured. These technologies have not been widely adopted as of yet, however, as they can be difficult to learn at first and require a degree of consideration for data (and metadata) management.

However, the U.S. government has begun to make semantic data available as a possible format for consumption of the government datasets they release on Data.gov[1], which demonstrates that the government is (somewhat) aware of the importance, or at least advantages, of cultivating semantic data. Not all Data.gov datasets are published in semantically robust formats, however, which points to a need for further effort to employ semantic frameworks within data.

Leveraging semantic web technologies and government data has various innovative and applicable benefits. The Tetherless World Constellation, a research group at RPI focused on semantic web research, has had prodigious success in applying these technologies to open government data (OGD) posted by Data.gov, via their Linking Open Government Data (LOGD) project [2]. They have succeeded in converting many OGD datasets to RDF, (a semantic data format), which they make available for public consumption, and even provide tools, such as a SPARQL endpoint, to allow automatic access of portions of the data by web services and computer programs. Even more impressive is their collection of demos highlighting how Semantic Web and OGD can be combined to perceive data, by using powerful visualization tools. In essence, they demonstrate that semantic web technologies can be applied effectively to OGD and can even improve its value as a semantic resource on the web of data.

This paper is organized as follows: Chapter 2 provides a historical review of literature detailing some of the problems and approaches explored in relation to linking data. Chapter 3 gives a brief background of the semantic technologies and concepts relevant to this paper. Chapter 4 explains the problem and motivation of interlinking Data.gov data, by U.S. states, including an identification of the major tasks involved. Chapter 5 introduces automated implicit linking as an approach to solving the problem and describes the implementation. Finally, Chapter 6 presents an evaluation of the system, with the results discussed in Chapter 7.

[1] Data.gov is the name of the website as well as the URL

2. LITERATURE REVIEW

Data linking can be conceived of as, in a way, an alignment of structured data. Whether the intent is to align data by values or dimensions, the modern approaches stem from earlier work done with schema matching. In several papers, Erhard Rahm and several other authors attempt to classify many of the techniques that have been used to map modern-day schema in relational databases. A survey conducted by Rahm and Bernstein in 2001 detailed the traditional approaches to automatic schema matching, and developed criteria to classify them based on the type (terminological, semantic, structural) and interpretation (external, semantic, syntactic) of the data they dealt with [4]. The use of natural language processes algorithms and techniques were found to be highly successful in matching terminological data within schemas, whereas graphical pattern comparison was useful for structural mapping [4]. A distinction is made between approximate and definite matching, especially when encountering linguistic or string encoded data. Edit distance and the utilization of taxonomies are considered to be the most useful when working with natural language strings [4].

Four years later, Shvaiko and Euzenat published a paper which more formally described the problem of schema matching, introducing a definition for what they termed *The Matching Problem* [7]. The matching problem can be, in essence, reduced to a conceptual mapping problem in the most general sense; it is the goal to determine whether element or pattern A can be matched to B. They do not place emphasis on the actual mapping operation, but instead focus on the importance of the inputs and outputs (referred to as mapping elements) of the system, as well as the necessity of confidence metrics as a means of verification [7]. In [5], Melnik, Do, and Rahm conducted a comparison of common schema matching evaluation systems in order to derive a definitive set of standard criteria that could be used to measure confidence of element and structural mappings within schemas/ontologies. At this time, schema and ontologies were still rooted in the entity-relationship model, typically used by relational databases, and had not yet expanded to graphically structured data. However, the systems which were developed based on this work demonstrated techniques which could be adapted in useful ways for other types of structural alignment. One of the systems they reviewed was the COMA++ schema/ontology matching system. COMA++ utilizes two major

strategies, (1) fragment-based matching (of elements or a part of the schema structure) and (2) reuse of previously determined mappings [8]. It also incorporates the idea of using a "pivot schema", a standard schema that can be used as basis of comparison to newly encountered schema [8]. Another system that offers a solution to the schema matching problem is sPLMap, which uses a probabilistic approach [6]. Oriented around Datalog, a logic-based language, sPLMap conducts matching by defining the mapping between two aspects of a schema in terms of a mapping rule, which it then attempts to validate using a set of previously discovered rules [6]. While both COMA++ and sPLMap present unique approaches to schema matching, what they have in common is the use of supplementary data to facilitate the mapping of elements (COMA++ uses previous matches and pivot schemas, sPLMap relies on a KB of logic rules). In later work, especially within computational linguistics focusing on entity resolution, it would be identified as crucial to have as much contextual support for data as possible, be it from foreign sources or linguistic data features themselves. In 2006, [9] studied the effect of including more contextual information associated with schemas to see whether more accurate mappings could be produced as a result. It was found that context and metadata revolving around a schema can also be used to augment evaluation metrics, which results in stronger and sounder mappings, an idea which was adapted by many semantic web and data linking researchers looking to cultivate links from literals contained in RDF (a metadata model used to structure data) data.

The intentions of data linking across the semantic web can be described by Tim Berners-Lee's notes on Linked Data [3], which generally state that web documents which mention resources should do so using URIs (for the resources). Unfortunately, as it stands, most of the documents on the Linked Open Data cloud do not contain URIs, and therefore are not truly linked data [12]. This is distressing as a lack of linked data undermines the entire purpose of the semantic web, and therefore many researchers have attempted to either promote resources or develop methods which would allow linked data to be easily published (lessening the burden on data managers) or derive linked data from existing sources. Recently, this task has fallen within the domain of computational linguistics, which seeks to resolve semantic entities from literal string values.

One area of research in which entity-resolution has been explored is computational linguistics. Entity resolution is essentially the derivation of a semantic entity from a literal value. Ultimately, entity resolution is about natural language disambiguation. In [13] and [15], Cucerzan and Pasca both agree that the internet can be treated as giant corpus or collection of human information that can be utilized to disambiguate and categorize literal terms. In [13], Cucerzan describes a disambiguation paradigm in which, utilizing Wikipedia, the surface forms (literal representations) of entities are identified in order to facilitate data association between discovered literals, which could match these surface forms, and entities. Cucerzan goes on to list what cues can be used to resolve a string to an entity, which are mappings to surface forms, tag categories and contextual information [13] which supports an entity. Pasca goes further than Cucerzan in [15] by trying to identify names and their associated categories, and then use this acquired knowledge to classify newly discovered terms (linking via classification). Either way, both admit that having as much knowledge as possible is highly beneficial to disambiguation. A survey done by Nadeau and Sekine, which examines some of the techniques used to recognize and classify named entities, corroborates the claim that context is important, but determines that it must be relevant for sufficient recognition of "proper names", especially in unsupervised learning (automatic) systems[16]. Relevance of supplementary information is determined by information retrieval and similarity measures, which are also techniques that can be applied to verification within named entity resolution [16]. Many of these measures and techniques are similar to those used in schema matching and have a basis in NLP and computational linguistics. In [17], Ratinov and Roth propose a few key decisions that must be addressed when constructing named entity recognition systems, one of which is "How to use external knowledge resources?". They describe a two-stage prediction method which applies a baseline named entity recognition system to data and then uses the resulting predictions to make more focused inferences about identified terms within data. Approaches such as these, which use linguistic disambiguation techniques and selectively chosen external supportive data to perform subsequent inferences of whether or not an entity identification is correct are what can potentially make named entity resolution systems very accurate.

In a paper published by Resnik which examines an attempt at disambiguation of natural language terms within a IS-A taxonomy by using semantic similarity (linguistic and taxonomic structural features shared by two terms), he notes that disambiguation relies heavily on natural language processing [10]. This is important because while the foundation of the semantic web is RDF and linked data will operate within this design space [18] (which is the focus of this paper), it will involve many of the challenges faced within Resnik's experiment with taxonomies, and it is apparent from his work that NLP is a required part of the solution to linking data. While Resnik's paper was published in 1998, a recent study into the use of semantic similarity in IS-A taxonomies (now part of the LOD cloud [23], see Appendix B) has demonstrated promising results to using NLP to link data [11]. Work published in [19] describes efforts to interlink music data from RDF datasets to other data sources on the LOD cloud and also redefines the mapping problem (similar to the previously mentioned schema matching problem) for the RDF space. This study uses other core semantic web technologies such as SPARQL to perform literal text matching across multiple datasets. Literal text matching, utilizing SPARQL, essentially allows one to query a store of datasets for a specific term, and then, if a specific dataset contains that term, acquire the associated URI, which is assumed to describe the entity that a string term refers to [19]. Interlinking is done primarily to DBpedia, a popular datasource on the LOD cloud which defines and provides URIs for millions of entities, mostly derived from Wikipedia [20]. The Linking Open Government Data (LOGD) project [14], which is hosted by the Tetherless World Constellation, has been doing research into how government data can be interlinked and externally linked to DBpedia. In [14], members of LOGD discuss a series of principles they use when promoting open government data to linked data and mention using DBpedia as a way to connect Data.gov datasets to linked open data. Sites like DBpedia are useful because it saves the trouble of data managers having to mint their own URIs and construct their own documents/ontologies for describing resources on the web by allowing them to simply link to DBpedia URIs instead. DBpedia is linked to a variety of major linked data sources on the web and is therefore a desirable corpus to link literals found in Data.gov datasets to by the system described in this paper.

3. RELEVANT SEMANTIC WEB BACKGROUND

The Semantic Web (SW) is a general term invented by Tim Berners-Lee to describe a network of data or human-readable pages, extended with machine-comprehensible semantics, which could be processed by computers automatically. The efforts to cultivate the Semantic Web are driven by frameworks, such as RDF and OWL, which allow data to be structured with semantically relevant metadata, such as descriptive annotations and links to other related resources. The topics discussed in this paper rely on some knowledge of the Semantic Web and a few of its underlying core technologies to explain the problem space. In the following sections of this chapter, the pertinent SW concepts, which apply to implicit data linking, will be presented in brief.

3.1 Relevant SW Technologies

3.1.1 RDF

Resource Description Framework (RDF) is the standard model used within the Semantic Web to structure information about resources, or entities[2], and uses XML syntax (RDF/XML) to represent statements about resources in a way that machines can process [21]. While RDF adopts XML's syntax, its data model is considered to be more graph-like as opposed to XML's usual tree-like conceptualization. An RDF datasets contains a list of statements, each which structures and encapsulates data within the overall RDF graph. An RDF statement is abstractly represented as a single graph arc, but it can also be serialized by using triple notation. The triple, which is the basic element of RDF, and can be conceived of as a 3-tuple consisting of a subject, predicate (or property[3]), and object [21]. Figure 3.1 demonstrates an example RDF statement as it would appear in a graphical representation and the associated triples in RDF/XML. An English interpretation of the second triple would be that the subject, *example.org/staffid/85740* (which is an entity) has a predicate, *name*, (also an entity) which has a value of "John Smith" (a literal). The object is the only one of the three that can be either a literal value (i.e., string, number, etc.) or an entity, as the differences between the first and second

[2] The terms 'entity' and 'resource' is interchangeable; for the duration of this paper, 'entity' will be the preferred term.
[3] The terms 'property' and 'predicate' are interchangeable, however predicate is used more frequently.

triples in Figure 3.1 demonstrate. An entity can be any abstract concept or thing, such as people, countries, the number one, etc. Entities are usually represented by a URI (Universal Resource Identifier), a specific type of URL which can be manufactured to identify, either locally or globally, a single entity [21]. A URI consists of two components: a namespace and a local name (everything that follows after the last forward-slash). While local names can sometimes be expressive, such as in Figure 3.1 with the predicate local names *age* and *name*, they can also be arbitrary, like *85740* from the second subject's local name.

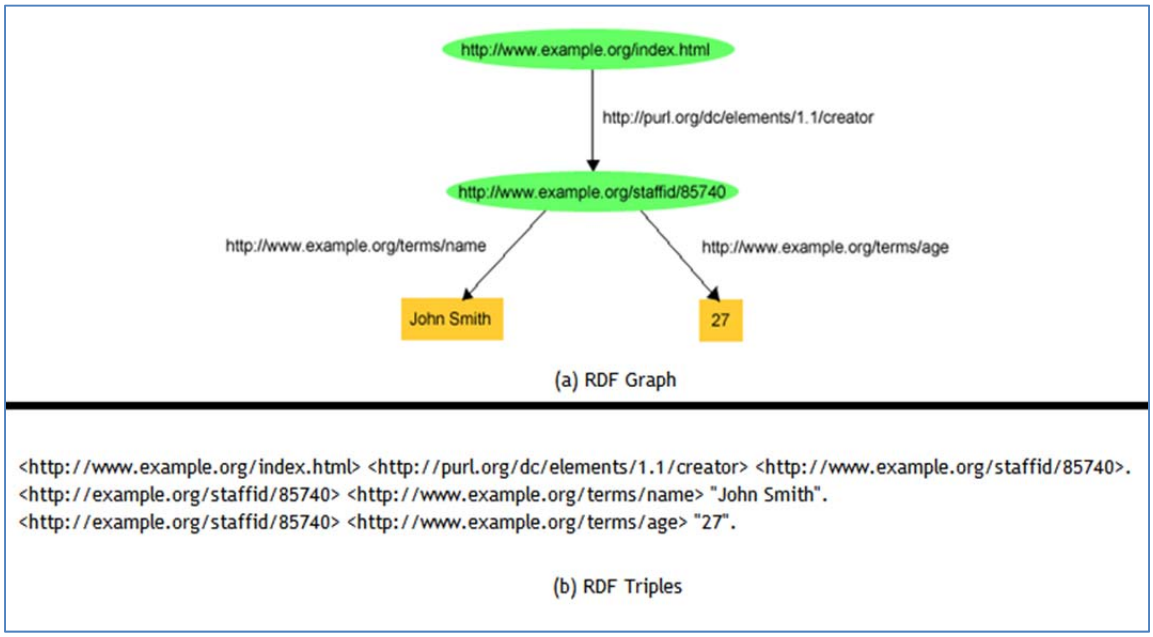


Figure 3.1: Example (a) RDF graph and (b) triple representation

The real benefit of URIs is that they can be dereferenced (followed) to acquire additional information about the entities they identify, themselves. The capacity to combine these URIs with human-readable annotations, all encapsulated within a framework which facilitates machine-readability, makes RDF a powerful tool to publish and interconnect data on the web.

3.1.2 SPARQL

There is a query language designed specifically to query and retrieve data over RDF graphs, known as SPARQL [22]. The syntax, as seen in an example query in Figure 3.2, is roughly similar to SQL in its use of the *select* and *where* clauses to define which data

should be selected given a set of selection parameters. However, whereas SQL operates over relational databases, where data is stored in a tabular arrangement, SPARQL traverses RDF graphs searching for patterns based on the provided parameters. In Figure 3.2, we can see a clause called *graph*, which is used in SPARQL to direct the query specifically at the graph encapsulated by that clause. There can be multiple instances of the *graph* clause which allows multiple graphs to be examined using a single SPARQL query. To be able to query RDF graphs, they must be loaded into a triple store, a special kind of database made to store RDF triples.

```
PREFIX conversion: <http://purl.org/twc/vocab/conversion/>
SELECT ?s ?p ?o
WHERE {
  GRAPH <http://logd.tw.rpi.edu/source/data-gov/dataset/249/version/1st-anniversary/conversion/raw/subset/sample>
  {
   ?s ?p ?o.
  }
}
```

Figure 3.2: Example of a SPARQL Query

The structure of a SPARQL query somewhat emulates the graph-like nature of the RDF datasets that they query. The select parameters (those of the form *?x*, where *x* is the name of the parameter) act simply as place holders for nodes and edges within the graph; other than that they have no relevance on their own. In Figure 3.2, we query for a single graph arc by searching for any node (*?s*) which has any predicate (*?p*) linking it to any other node (*?o*). From the query results, in Figure 3.3, we see can see which entities and literals have been returned by this "fish-net" query. The query could be expanded to include more graph patterns attached to *?s* to find more specific results. Furthermore, for any objects that contained entities (URIs), we could expand the query to explore those object nodes the same in the same way as was done for the subject node, by searching for any graph features it may have connected to it. This is one way in which an entity can be dereferenced (followed) to acquire more information. There are many features of SPARQL which allow it to query RDF graphs with varying levels of specificity, thus making it a useful tool when working with semantic data.

SPARQLer Query Results

s	p	o
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/ancillary_or_other_use>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/as_a_chemical_processing_aid>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/chemical_name>	<http://www.w3.org/2000/01/rdf-schema#label>	"CHEMICAL NAME"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/classification>	<http://www.w3.org/2000/01/rdf-schema#label>	"CLASSIFICATION"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/date_signed>	<http://www.w3.org/2000/01/rdf-schema#label>	"DATE SIGNED"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/db_nr_b>	<http://www.w3.org/2000/01/rdf-schema#label>	"DB NR B"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/db_nr_a>	<http://www.w3.org/2000/01/rdf-schema#label>	"DB NR A"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/dioxin_distribution_1>	<http://www.w3.org/2000/01/rdf-schema#label>	"DIOXIN DISTRIBUTION 1"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/dioxin_distribution_10>	<http://www.w3.org/2000/01/rdf-schema#label>	"DIOXIN DISTRIBUTION 10"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/dioxin_distribution_11>	<http://www.w3.org/2000/01/rdf-schema#label>	"DIOXIN DISTRIBUTION 11"
<http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/dioxin_distribution_12>	<http://www.w3.org/2000/01/rdf-schema#label>	"DIOXIN DISTRIBUTION 12"

Figure 3.3: Results from sample SPARQL Query

3.2 Linked Data

Tim Berners-Lee, founder of the Semantic Web, coined the term "Linked Data" and defined it to mean, essentially, data which incorporates URI references for named entities and other relevant datasets, when applicable [3]. The data could contain literal values as metadata annotations to refer to external resources, but in order for data to be truly linked, it would require out-going links to be present. The use of external URIs is what makes a dataset more valuable as a resource on the semantic web, because it allows machines to easily traverse the network of URIs to acquire or discover new information. The Linked Open Data cloud [23], found in Appendix B, is a collection of semantic data and entity sources and depicts the current state of linked open data on the web. DBpedia [25] is one of the most popular linked data nexuses, primarily because it is a project which converts articles from Wikipedia to semantic web entities, which can be openly linked to from other data sources by referencing DBpedia's minted URIs [20]. Wikipedia comprises a large corpus of human knowledge, and as a semantic resource through DBpedia, provides a wide array of descriptive semantic entities for use on the web, allowing data to be linked in a way that is machine-accessible. However, this relies on data publishers to (1) structure their data in a semantically conscious way and (2) incorporate these existing URIs for semantic entities in the data.

3.3 Combining Semantic Web and Open Government Data (OGD)

The Tetherless World Constellation (TWC) at RPI is a research group focused on semantic technologies and runs a project, called the LOGD (Linking Open Government Data) project, which looks at ways to apply them to linking open government data (OGD). One of the major problems with OGD, especially within Data.gov datasets, is its lack of data semantics and links to both other relevant data and popular entities (i.e., links to DBpedia) on the semantic web. One of their most notable accomplishments has been to convert open government data released by Data.gov from common data formats, such as CSV, to the semantically rich and machine-readable RDF format. They make these converted datasets available to the public on their site, which also provides tutorials on how to use converted data. Furthermore, they maintain a triple store on which they host converted data and have set up their own SPARQL endpoint to allow it to be queried. They have multiple visualizations and demos which explore the potential of how OGD and semantics can be combined effectively. Much of LOGD's previous efforts and publicly available resources, particularly the converted government datasets, have been vital to establishing the problem and approach presented in this paper, which concentrate on linking between government data and semantic entities.

4. THE PROBLEM – LINKING OPEN GOVERNMENT DATA BY U.S. STATES AND TERRITORIES

4.1 Motivation and Problem Description

As touched on in the previous section, datasets from Data.gov can contain relevant information to any number of categories (i.e., environment, budget, etc.) or entities (i.e., cities, people, etc.) which are commonly known. However, these relations are often occluded due to a lack of URIs within Data.gov datasets, which should be used to disambiguate references to entities and categories. Exclusion of dereferencible URIs in data is not just an issue for automated agents that rely on them to gain extended knowledge of mentioned entities, but also for human users looking to effectively and accurately use data. Insight into how data is structured, such as which column names or properties refer to desired categories, is also necessary for users to decide along which mutual dimensions data can be aligned. From a consumer standpoint, having a reliable way to peer into the contents of data, without actually having to review the raw bits themselves, would be useful and cost effective for mashing up data accurately. This perspective could be obtained by amending Data.gov datasets with URIs for named entities and associating the predicates attached to them, thereby allowing users, human and machine, to assess the contents of a dataset quickly and determine how best to combine it with other datasets. The ability to integrate, or mash up, different data sources is powerful and desirable, especially if the sources contribute diverse categorical knowledge, because it enhances the perception of the collective data, allowing a user to derive a better understanding of it, as opposed to just examining a solitary dataset. Therefore, it is imperative that URIs for named entities be incorporated into Data.gov datasets to give users the ability to mash up data.

This paper examines the specific problem of linking Data.gov datasets to any U.S. state or territory entities that they may contain. Focusing the problem on a specific entity (U.S. states and territories) was a strategic decision made to limit the scope of the much larger data linking problem, thus reducing it to the much more manageable state-linking problem. Primarily, there are three main questions posed by the state-linking problem:

1. Given a Data.gov dataset, which U.S. states or territories are mentioned within it?
2. Which predicates within the dataset refer exclusively to U.S. states or territories (a.k.a. are state-related)?
3. Can an automated solution which answers the above questions be generated?

It is assumed that the answers to these questions are prerequisite for any adequate solution to the problem. However, in attempting to explore possible solutions for them, many inherent obstacles to linking data are revealed. These obstacles, which can be broken down into subsequent tasks within the problem space, will be discussed in full in the next section. It is our belief that by addressing each of these obstacles within our approach, we can construct a unified solution which can be utilized by a generalizable method for solving the overall problem of data linking for open government data in future works.

4.2 Tasks within the Problem Space

4.2.1 Entity Resolution

Within the semantic web, it is generally accepted that a URI be used to describe a single entity. There is a strict many to one (1..* → 1..1) relationship between URIs and named entities. There can be multiple URIs which describe a single entity, but each one must be designated to a single entity alone. The exclusivity of URIs makes them advantageous over string literal for identifying entities and also for disambiguating data. If a predicate contains an object URI, it can be determined with absolute certainty what the object is referring to by simply dereferencing the URI to access information regarding the entity. This is not the case for literals. First of all, machines do not have the capability to inherently disambiguate strings, without some kind of context or knowledge base to provide them with the meaning. Secondly, string literals can have a many to many relationship (1..* → 1..*) with named entities, thus making them non-unique for identification purposes. For example, the string literal "New York" can refer to the well-known named entities New York City, New York State, or New York County and perhaps others as well (i.e., streets, towns, etc). Likewise, the entity New York State can be represented by "NY" (postal code), "36" (FIPS code) and just "New

York". The differences in literal representations can also be more nuanced, such as capitalization ("NEW YORK"). From a semantic data linking perspective, how would a machine be able to disambiguate the word "New York"? The answer is it wouldn't, unless it had access to a large corpus of words and definitions, like an encyclopedia, and it would also require the ability to perform [expensive] NLP methods to consume the word. Given the amorphous nature of textual labels, it is very challenging to determine which literal representations can be accurately attributed to which state entities.

Data.gov datasets use string literal references (within RDF objects) rather than URIs to denote U.S. states or territories. The concept of keyword search comes to mind as a possible solution to searching for state-related entries within Data.gov, given its popular success with locating data from string parameters. However, each agency has their own way of depicting states and territories, meaning there is no way to find all state information by performing a keyword search over the data. Searching for a keyword, like "New York", would return entries that used that keyword to identify New York. If the purpose was to find data for New York State, only data entries which used the "New York" keyword to identify the state would be returned. Entries which used a variant but legitimate string identifier, such as "NY", would not be returned, and so only a portion of the data would be available to the user via keyword search. Perhaps then a federated keyword search which looks through Data.gov datasets for every representation of New York could be adapted as a solution? This is also found to be not quite adequate when one considers the semantic garbage values that are returned from raw keyword search. Semantic garbage values have content which matches the keywords one is searching for, but refer to semantically different entities than the one which is intended to found. For example, searching for "New York" or "NY" could cause search to return entries for New York city as well, especially in Data.gov data which contains data for states and cities[4]. Therefore, a more robust method which has the capacity to semantically evaluate string literals before making any assertions as to the entities which they reference must be part of a solution to the state-linking problem.

[4] There would be even more noise returned if searching by FIPS codes and other numeric identifications for states, but a semantically conscious solution would probably still be adequate to resolve them.

4.2.2 Verification

Entity resolution requires some justification or verification to ensure that it is appropriate. To make the blind assertion that some literal is equivalent to an entity is meaningless unless there are sound reasons for doing so. Generally, the entity resolution process relies profoundly on context and background knowledge to make any determinations. Recognizing the appropriate metrics and methods required to perform precise literal-entity mapping is challenging. This is especially so when it applies to Data.gov datasets, which have very little contextual information to draw upon for verification. Knowing how to apply context effectively is therefore key for successful authentication of the assertions made by entity resolution. Constructing a way to utilize whatever context is available (i.e., values attached to predicates, predicate local names, etc.) in an apposite manner which positively contributes to the assessment of the state-relatedness of something, either literal or predicate, is a vital task which must be solved before an approach to the state-linking problem for Data.gov data is undertaken.

4.2.3 Automation

The converted Data.gov datasets can contain anywhere from a couple thousand to millions of triple. Even dealing with only a couple datasets could mean having to look over hundreds of millions to billions of triples, making it less practical for users to wade through the raw data and analyze it manually to get a sense of what interesting entities or content it contains. A better solution would be to construct an autonomous agent which could investigate and report back any interesting features that a dataset might have which could be valuable or relevant to users. The fact that natural language processing for machines is still a very difficult problem makes this solution hard to implement. Naturally, NLP and computational linguistic techniques would have to be adopted by such a system in order for it to derive meaning from text. The best techniques are those that draw upon as much contextual and background knowledge as possible, but these are also very computation heavy and expensive to run, depending on the size of the knowledge base. In order to construct a system which is capable of linking string literals to entities, a balance between using powerful semantic processing algorithms and

scalability will have to be maintained in order for such a system to be beneficial from a user-standpoint.

5. AUTOMATED IMPLICIT LINKING APPROACH

An approach to solving the state-linking problem for Data.gov datasets takes the form of the implicit linking system, which automatically generates links between U.S. state and territory URIs (derived from the string literal values in the original dataset) and predicates which are deemed to be state-related by the system. These links are supplemented by weights which convey a measure of how likely they are to be accurate. Weight measures are calculated by various natural language processing (NLP)-based heuristics which attempt to verify whether a predicate is state-related once it is found to be attached to a literal string which has been resolved to an entity. The system outputs implicit link sets, which can be used to complement the original Data.gov datasets that they are derived from by offering users a way to query for state-related content directly without having to manually inspect the raw data.

5.1 Tools and Technologies Used for Implementation

The core system was implemented in JAVA and developed on the open source Eclipse IDE. The Jena API libraries [24] were incorporated into the into the standard JAVA libraries to add support for semantic technologies, specifically for RDF and SPARQL. Jena API allowed the system to issue queries to SPARQL endpoints and download the results for processing. The LOGD project provided much of the data infrastructure and semantic technology resources used by the implicit linking system. They hosted a SPARQL endpoint, which was pointed at an RDF triple store (a specific type of database designed to store large numbers of triples), giving the system the capability to query for desired data and receive the results. Data.gov datasets were converted to RDF by LOGD and hosted on the web for public consumption. Not all the full Data.gov datasets are loaded into their triple store (therefore, they were non-accessible via their SPARQL endpoint). However LOGD does have sample datasets, taken from the larger Data.gov datasets, available within their triple store for queried access. These sample sets were used to test and perfect the implicit linking system throughout development, and later to evaluate it.

5.2 System Description

A workflow diagram of the implicit linking system can be found in Appendix C. Essentially, there are two main system components: Entity Resolver and Predicate Processor. Upon receiving a Data.gov dataset as input, the Entity Resolver attempts to map string literals found in the dataset to URIs from a mapping set (catalog of named entities). The predicates attached to all successfully mapped string literals are then fed to the Predicate Processor for verification of state-related status. Using a number of heuristics, the Predicate Processor outputs a likelihood measure of whether a predicate is related exclusively to state entities. A new implicit linking set is then constructed containing associations, or links, between the input dataset, its state-related predicates, and the likelihood measure for each predicate.

5.2.1 Input and Output

The system takes in two inputs: (1) the graph name of a Data.gov dataset, converted to RDF format and (2) a list of user-specified search terms. An example of a converted Data.gov dataset is shown in Figure 5.1, in RDF/XML format[5]. Only the graph name to which the dataset is loaded into a triple store is taken in, not the actual dataset itself. The system does not interact directly with the Data.gov dataset, but instead queries a SPARQL endpoint, (attached to a triple store where the dataset is loaded), to acquire the parts of the dataset that the system requires. The relevant parts of the data examined by this system are the objects literal values, which are attached to predicates (Figure 5.1 depicts them as values encapsulated by property tags). The literal values and predicate names are primarily used to identify and verify the existence of state entities and state-related predicates. The second input consists of search terms provided by the user. Many of the predicate names in Data.gov datasets contain keywords which usually can be used to determine whether a predicate is state-related. In Figure 5.1, if you locate the predicate name (tag) *ha_state*, you will see that it does indeed refer to a U.S. state, in this case "New York". If one were to do a manual survey of all the predicate names in

[5] The Data.gov datasets converted by LOGD are now in Turtle format, a variant serialization of RDF. Figure 3.1 was used to demonstrate the dataset in a layout (XML) which may be more recognizable to those who are not familiar with RDF in general. Differences between RDF/XML and Turtle are purely syntactical and do not affect the data itself.

all Data.gov datasets, they would likely find that most, if not all, of the predicate names which contain the word "state" happen to be state-related. This is an assumption made based on the frequency and consistency of encountered predicate names which have had the keyword "state" and been state-related. Given that RDF Data.gov datasets may boast upward of a million triples, this assumption is used to quickly pass over obvious cases in which state-related predicates can be identified by the inclusion of specific keywords in their local names.

```
<rdf:Description rdf:about="#entry387">
  <fo_name>Buffalo Hub Office</fo_name>
  <ha_low_rent_size_category>No units</ha_low_rent_size_category>
  <rdf:type rdf:resource="http://data-gov.tw.rpi.edu/2009/data-gov-twc.rdf#DataEntry"/>
  <number_of_low_rent_projects_under_mgmt>0</number_of_low_rent_projects_under_mgmt>
  <number_of_section_8_units>49</number_of_section_8_units>
  <ha_state>New York</ha_state>
  <number_of_low_rent_units_under_mgmt>0</number_of_low_rent_units_under_mgmt>
  <ha_name>Whitestown, Town</ha_name>
  <number_of_low_rent_units_under_development>0</number_of_low_rent_units_under_development>
  <hub_code>2HBUF</hub_code>
  <fo_code>2CPH</fo_code>
  <ha_section_8_size_category>Very Small (1 - 49)</ha_section_8_size_category>
  <ha_county>Oneida</ha_county>
  <ha_fye>30-Sep</ha_fye>
  <region_name>New York/New Jersey</region_name>
  <formal_ha_name>Town of Whitestown</formal_ha_name>
  <ha_congress_district>23</ha_congress_district>
  <hub_name>Buffalo Hub</hub_name>
  <ha_city>Utica</ha_city>
  <region_code>2</region_code>
  <number_of_section_8_projects>1</number_of_section_8_projects>
  <number_of_low_rent_projects_under_development>0</number_of_low_rent_projects_under_development>
  <ha_program_type>Section 8</ha_program_type>
  <ha_combined_size_category>Very Small (1 - 49)</ha_combined_size_category>
  <ha_code>NY542</ha_code>
</rdf:Description>
```

Figure 5.1: Excerpt from Data.gov Dataset 1464, converted to RDF by LOGD

The goal of the system is to produce some output which identifies state-related entities and predicates within the input Data.gov dataset, and generate links between them and dereferencible URIs for the entities (in this case from DBpedia). One approach would be to append the appropriate URIs to literal entity representations wherever they occur in the original dataset, that is, to modify the actual dataset. Since the URIs were not contained in the dataset to begin with, however, it appears improper to treat the addition of missing URIs the same as if they were explicitly declared to

begin with. The linking system relies almost entirely on implicit linking. The linking is implicit because it uses NLP-based heuristics to derive information from existing literal string content within a dataset in order to identify state entities and state-related predicates. Also, there is no human verification of the results, and so therefore it is more appropriate to say that the system implies links (with some certainty). To give users a non-invasive way of incorporating the links found by the system alongside their existing data, the system outputs an implicit link file. This implicit link file would potentially be hosted in conjunction with the original dataset so as to allow one to query it and receive all the original dataset's state-related content. Figure 5.2 shows an excerpt from an example output file. As seen, each entry (began by the *rdf:Description* RDF/XML tag) contains an association (link) between the original dataset and a predicate (from the original dataset), the URI of a state or territory entity (*rdf:object*), and a weight (*ov:weight*) which is the system's confidence measure of how likely the connotation is to be accurate. From Figure 5.2, the second entry can be seen as having a weight value of 0.0615. All weights are standardized to be between 0 and 1, so therefore 0.0615 is actually very low, implying a very low likelihood that the predicate *facility_county* (local name) mentions the state of Texas.

```
    "http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/mailing_state"/>
    <rdf:object rdf:resource="http://data-gov.tw.rpi.edu/vocab/Idaho"/>
    <rdf:dataset rdf:resource=
    "http://logd.tw.rpi.edu/source/data-gov/dataset/249/version/1st-anniversary/co
    nversion/raw/subset/sample"/>
    <dcterms:identifier>"249"</dcterms:identifier>
  </rdf:Description>
  <rdf:Description rdf:about="4abc7fd2b06430baf44abc7a777affb0">
    <ov:weight>"0.0615"</ov:weight>
    <rdf:predicate rdf:resource=
    "http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/facility_county"
    />
    <rdf:object rdf:resource="http://data-gov.tw.rpi.edu/vocab/Texas"/>
    <rdf:dataset rdf:resource=
    "http://logd.tw.rpi.edu/source/data-gov/dataset/249/version/1st-anniversary/co
    nversion/raw/subset/sample"/>
    <dcterms:identifier>"249"</dcterms:identifier>
  </rdf:Description>
```

Figure 5.2: Excerpt from implicit link dataset output example

The details of how weights are calculated are discussed in a later section, but a preliminary analysis based on the predicate name alone makes it clear the predicate is

likely referring to a Texas county (possibly Texas County, Oklahoma). Even though the system makes this determination, it performs no filter operations on predicates and allows them to be included in the results despite how low the likelihood weight is calculated to be. This is intentional, as we felt it best to give the user the option to set the weight threshold. Doing any kind of pre-filtering would be placing too much faith in the system's ability to discern state-related predicates from non-state-related predicates. Even with 100% certainty, it is only making such determinations based on data derived from NLP processing, which, being a machine, it currently has limited capacity to do. It is more suitable to regard the links outputted by the system as suggestions, and to use them as such.

5.2.2 Entity Resolver

The Entity Resolver addresses the entity resolution task (introduced in Chapter 4) by attempting to identify literal string values within Data.gov datasets which could potentially be promoted to state entities. As discussed in the literature review, entity resolution is a hard problem which typically relies heavily on computational linguistics and natural language processing to extract any kind of semantics from literal strings. To reign in the scope of the state-linking problem, a literal-entity mapping method is used which bypasses the need for NLP processing. Literal-entity mapping, in the basic sense, requires two components: (1) a string and (2) a mapping set.

```
<swivt:Subject rdf:about="&wiki;Arizona">
    <rdfs:label>Arizona</rdfs:label>
    <swivt:page rdf:resource="&wikiurl;Arizona"/>
    <rdfs:isDefinedBy rdf:resource="http://data-gov.tw.rpi.edu/vocab.php?instance=Arizona"/>
    <rdf:type rdf:resource="&wiki;c/State_of_the_U.S."/>
    <dcterms:modified>2010-2-26</dcterms:modified>
    <property:Dgtwc-3Aabbreviation rdf:resource="&wiki;AZ"/>
    <foaf:name>Arizona</foaf:name>
    <owl:sameAs rdf:resource="http://sws.geonames.org/5551752/"/>
    <owl:sameAs rdf:resource="http://dbpedia.org/resource/Arizona"/>
    <skos:altLabel>Arizona</skos:altLabel>
    <skos:altLabel>arizona</skos:altLabel>
    <skos:altLabel>ARIZONA</skos:altLabel>
    <skos:altLabel>AZ</skos:altLabel>
    <skos:altLabel>az</skos:altLabel>
</swivt:Subject>
```

Figure 5.3: Excerpt for 'Arizona' from an example mapping dataset for U.S. states and territories

A mapping set is essentially a catalog of statements which associate various string representations or labels of a named entity (i.e., a U.S. state) and one or several URIs which can be used to describe it. These labels can be any type of identifier generally used to describe a state, such as postal or FIPS code or state name but could also be something more nuanced, like an ID specifically used by the Department of Labor. To reign in the complexity of identifying states and territories, the system only concentrates on those representations provided by the mapping set and does no more extraction or inference beyond that. From the example in Figure 5.3, one can see that looks very similar to the Data.gov dataset example from Figure 5.1. The difference between a regular Data.gov dataset and a mapping dataset is that the latter was manufactured for the express purpose of associating entity URIs with literal string representations of those entities. The entry for Arizona in Figure 5.3 pairs URIs for the Arizona entity from DBpedia and Geonames with *skos:altLabel*s (used for alternative labels as defined by the *skos* vocabulary) which list different string labels frequently seen to refer to the state of Arizona. These alternative labels help to bridge the divide between verified entity URIs and [somewhat ambiguous] string labels and are crucial for literal-entity mapping. Currently, the mapping set graph name, which is hosted by LOGD, is hardcoded into the Entity Resolver.

Literal-entity mapping is performed by taking a literal string and finding a matching value within this catalog. The assumption is made that if a string match is found, (via the transitive property) the input string may be mapped to the URI of the matching string in the mapping set, thus promoting the input string to a named entity. Within the system, literal-entity mapping is performed by leveraging a SPARQL query to find matching literal values between the input Data.gov set and the mapping set (see Appendix A.1) and returning the respective URIs. Thus, the Entity Resolver component sets the baseline for which predicates are potentially state-related by identifying possible U.S. states/territories within a Data.gov dataset and returning the attached predicates (attached to the object literals from the dataset, not the mapping set) to be fed through the Predicate Processor for verification.

5.2.3 Predicate Processor

The system depends greatly on the Predicate Processor to verify that the candidate predicates discovered by the Entity Resolver really are state-related, and don't just contain a few state literals (which may not actually refer to states). The primary function of the Predicate Processor is to use a set of NLP-based heuristics to process features of predicates, such as [local] name and analysis of values attached to a predicate. The goal is to produce a confidence measure which can be used to assess the likelihood that a predicate refers exclusively to U.S. states or territories. The Predicate Processor is not a filter; it does not disqualify predicates but merely adjusts their state-related likelihood measures. To generate an overall weight for each predicate, the Predicate Processor relies on a heuristic approach to make guesses on the likelihood of a predicate's state-relatedness. This is because it is far too difficult, and possibly presumptuous, to make a legitimate and sound judgment based on the ambiguous nature of a predicate's textual context (i.e., literal values, name, etc.), which is the only supporting information the system relies on. The outputs of the heuristics are standardized to be between 0 and 1, and are designed to operate independently of each other though they all contribute to one final output measure, the final weight. This is to allow the possibility of including new heuristics in future work.

There are three heuristics, which operate sequentially, used to verify a predicate computation:

1.) Bag of words – This heuristic is used to examine the makeup of object values attached to a predicate, which has been returned by literal-entity mapping from an input Data.gov dataset. A SPARQL query is issued to return all the object values of a given predicate without any filtering parameters. An example SPARQL query and results can be found in Appendix A.2. The set of objects literals captured by the Entity Resolver for a predicate is compared to the set of all object literals returned by this query. If both sets contain the same elements, then it is concluded that a predicate only contains object literals referring to states, and thus the case for asserting the predicate as state-related is strengthened. If the sets are the dissimilar, a measure of how (dis)similar the two sets are would be returned. This is calculated as the ratio of the size of the set of state literals attached to the predicate to the size

of all predicate object values. The ratio is returned as the likelihood, as judged by this heuristic, that an input predicate is state-related.

2.) String equality – Probably the simplest heuristic, this method is the first to focus on the local name of a predicate as a means of verifying a predicate's state-relatedness. The local name is tokenized into a list of keywords (separated by spaces or underscore characters within the name itself), which are compared to every term within the inputted list of user-specified search words. The heuristic has a binary return value; if a match is found, a 1 is returned, and if no match is found once the search term list is exhausted, the output is 0. The heuristic is primarily used to capture the obvious cases, which are predicates within Data.gov datasets which contain specific keywords which qualify them as being related to a particular entity. An example would be a predicate which contains the keyword "state" or "fips", which, from what we know about Data.gov predicates, usually (if not always) qualifies a predicate as being related to U.S. states and territories. If this algorithm meets with success, it is a given that using the next heuristic, edit distance, is unnecessary, which saves on execution time since edit distance can be more expensive to run.

3.) Levenshtein Distance (edit distance), with prefix filtering – The final heuristic in the arsenal is a last ditch method to finding whether or not a term within the search term list can be derived from a tokenized predicate local name through manipulation. A tokenized value is iteratively transformed into a target search term, by adding, removing or replacing individual characters within the string. The number of operations required to complete this transformation is known as the edit distance. The similarity between the two strings can be calculated as the ratio of the edit distance over the length of the longest of the strings, subtracted from 1. If the edit distance is 0, the strings are the same and a similarity value of 1 is returned[6], otherwise a similarity measure as described previously is returned. Since edit distance is an expensive algorithm to run, especially with longer words, prefix filtering was enabled to limit comparisons to only those words which had like

[6] A value of 1 is returned automatically on behalf of the Levenshtein heuristic if the second heuristic returns true, since it is unnecessary to calculate the similarity of two strings which are the same.

features, specifically the same prefix. The prefix constraint was limited to the first letter of each word, and was useful in clearing up additional processing by only focusing on very likely cases. This type of approximate string comparison is considered necessary in order to handle cases when the predicate name contains word fragments which may be abbreviated forms of the search terms. Examples of what is meant by abbreviated forms (for predicate local names which include the word "state") can be seen in Figure 5.4.

1	http://logd.tw.rpi.edu/source/data-gov/dataset/1051/vocab/raw/state_name
2	http://logd.tw.rpi.edu/source/data-gov/dataset/1374/vocab/raw/state
3	http://logd.tw.rpi.edu/source/data-gov/dataset/1287/vocab/raw/st
4	http://logd.tw.rpi.edu/source/data-gov/dataset/1066/vocab/raw/state_code
5	http://logd.tw.rpi.edu/source/data-gov/dataset/623/vocab/raw/stabr
6	http://logd.tw.rpi.edu/source/data-gov/dataset/1932/vocab/raw/sta

Figure 5.4: Example of Data.gov predicates which contain different forms of the word "state"

5.2.4 Heuristic Weights

Each heuristic within the Predicate Processor computes a portion of the overall weight metric used to determine the likelihood that a predicate is related to U.S. states or territories. The final weight is simply a sum of all three heuristic weights, normalized to a value between 0 and 1, as shown in this equation:

$$Weight_{final} = \alpha(Heuristic_1) + \beta(Heuristic_2) + \gamma(Heuristic_3)$$

$$where, \alpha + \beta + \gamma = 1.0$$

This was initially done using a distribution of the weights to maximize the impact of heuristics which were believed to be more important. During preliminary tests, Bag of Words was given a larger impact given that it most actively determined whether or not a predicate was state-related by checking that it contained only state values. The subsequent heuristics, utilizing the predicate local name to verify state-relatedness, were deemed less important considering that it was a possibility for a predicate to not have any qualifying keywords or language features within its name, and yet still be state-related. In the next chapter, evaluations of the system will be carried out to test for an optimal distribution which maximizes the precision of the system.

6. EVALUATION

6.1 Experiment Set-up

An experiment was run using the system to observe its performance and to find the optimal values for the distribution weight of each heuristic and threshold value for the final combined weight. Complete RDF Data.gov datasets were unavailable as the memory required for LOGD did not have the capacity to load every dataset into its triple store. To acquire the diversity of Data.gov data and also to scale down the problem[7], samples were taken from 203 converted datasets. These sample datasets consisted of the first couple dozen entries from the original datasets (just under 1000 triples each). The LOGD project hosted these samples in triple store, allowing the implicit linking system to query them via their SPARQL endpoint. The mapping set used to perform entity resolution was taken from LOGD's Instance Hub project, which provides a dataset containing all U.S. state and territory entities, with literal representations linked to DBpedia URIs. The search terms for the system's predicate processor were hardcoded and consisted of the following terms: "*state*" and "*fips*". The evaluation for the implicit linking system was run within Eclipse on a Lenovo ThinkPad T410 running Windows 7 64bit OS.

The results of the experiment were evaluated using precision and recall measures. Precision is the ratio of correct (state-related) predicates returned within the results over the total number of all results returned by the system. Recall is the ratio of correct predicates over the number of predicates that exist within the gold standard. A precision and recall value of 1.0 is desired as it means that the system is able to return all state-related predicates without any garbage values. The gold standard was manually constructed to contain all the state-related predicates and used to compare the predicates returned by the system when calculating the evaluation measures. A total of 382 unique state-related predicates were discovered within the 203 sample datasets.

Parameter search was performed to find both the distribution of weights for the heuristics and the threshold value which maximized the precision and recall. The

[7] The purpose of the experiment was not to test the scalability of the implicit linking system and no considerations were given to this during the implementation.

distribution of weights refers to the importance percentage placed on the result returned by each heuristic during the final weight calculation, which is the sum of all the heuristic results. It was hypothesized that the bag of words heuristic should be given account for a greater percentage of the final weight calculated for each predicate than the other two. Likewise, determining how each individual heuristic performed was also of interest, as it was assumed that the combination of all three heuristics would be needed to achieve optimum precision.

6.2 Results

6.2.1 Analysis of Weight Distribution and Threshold Constraints

Utilizing parameter search, the optimal threshold was calculated by looking for the maximum precision and maximum recall. Preference was given to precision, however, as having accurate values returned by the system was deemed to be more important than recall.

Table 6.1: Maximum Precision and Recall values for each predicate weight threshold

Threshold	Max Precision	Max Recall
0.0	0.30222	1.0
0.1	0.99705	1.0
0.2	0.99725	1.0
0.3	0.99725	1.0
0.4	0.99733	1.0
0.5	0.99735	0.99746
0.6	0.99733	0.98953
0.7	1.0	0.96859
0.8	1.0	0.90838
0.9	1.0	0.88482
1.0	1.0	0.88482

As seen in Table 6.1, the maximum precision which can be obtained is 1.0, which is considered perfect accuracy and the highest recall available, given this value, is 0.96859. These globally optimum values are achieved with a threshold of 0.7 applied to the final weight results of each predicate returned by the system to filter them. It should be noted that the precision and recall pairs reported in the table are not associated with each other; that is, the precision and recall values report the maximum values possible for a range of weight distributions applied to the heuristics, which are reported in Table 6.2.

Table 6.2: Maximum Precision and Recall for each heuristic as a function of percentage of final weight

Threshold = 0.7	**Bag of Word Heuristic**		**String Match Heuristic**		**Edit Distance Heuristic**	
Percentage of final weight (α, β, γ)	*Max Precision*	*Max Recall*	*Max Precision*	*Max Recall*	*Max Precision*	*Max Recall*
0.0	0.99706	0.88743	1.00000	0.96859	1.00000	0.94764
0.1	0.99708	0.89267	1.00000	0.96859	1.00000	0.94764
0.2	0.99708	0.89267	1.00000	0.94764	1.00000	0.94764
0.3	0.99725	0.94764	1.00000	0.94764	1.00000	0.94764
0.4	0.99725	0.95026	1.00000	0.88482	1.00000	0.96859
0.5	1.00000	0.96597	1.00000	0.88482	1.00000	0.96597
0.6	1.00000	0.96859	0.99705	0.88482	0.99725	0.95026
0.7	0.69349	0.94764	0.99705	0.88482	0.99725	0.94764
0.8	0.68893	0.94503	0.99705	0.88482	0.99708	0.89267
0.9	0.66913	0.94764	0.99705	0.88482	0.99708	0.89267
1.0	0.64935	0.91623	0.99705	0.88482	0.99706	0.88743

The values in Table 6.2 were acquired by using the optimal threshold value of 0.7. The table contains the values of the maximum precision and recall as a result of incremental changes in each heuristic's associated distribution weight (Bag of words - α, String match - β, Edit Distance – γ). Note that a recall of 1.0 is not possible within the range of values constrained by the threshold value, which is acceptable since the goal is to maximize precision. The columns within the table are non-associative, the percentage of final weight values only apply to each heuristic respectively.

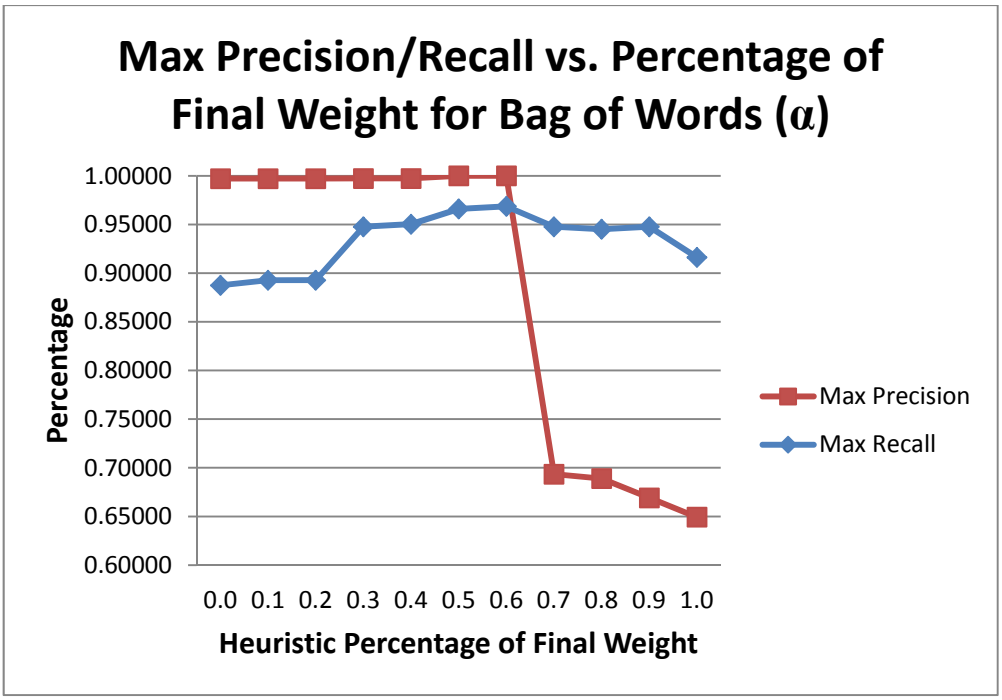


Figure 6.1: Graph of Max Precision and Recall vs. the percentage of the final weight for bag of words heuristic

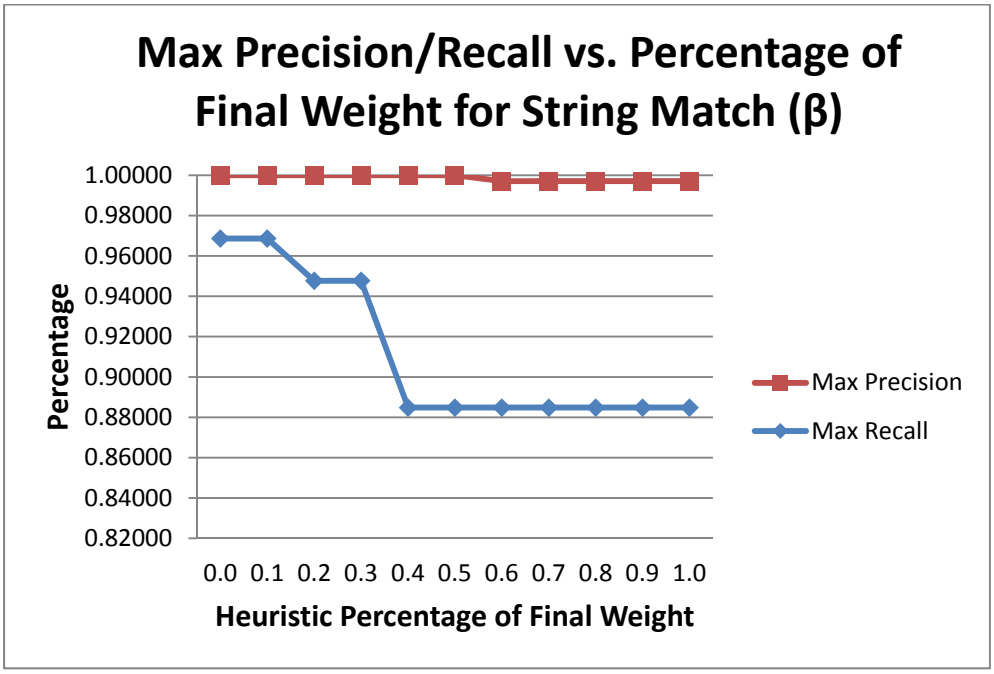


Figure 6.2: Graph of Max Precision and Recall vs. the percentage of the final weight for string match heuristic

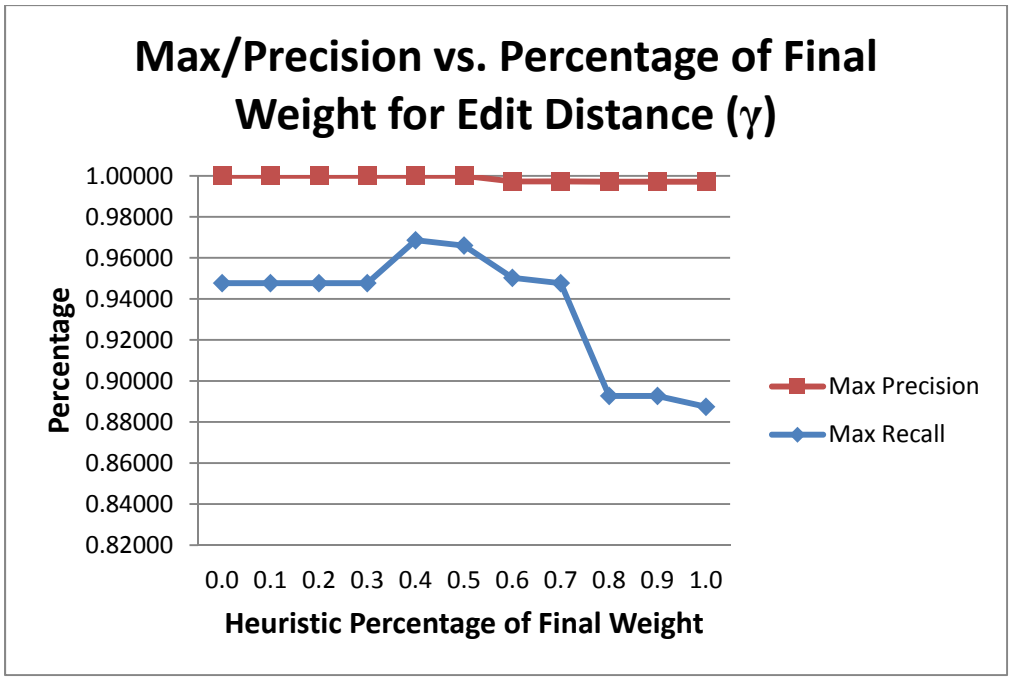


Figure 6.3: Graph of Max Precision and Recall vs. the percentage of the final weight for edit distance heuristic

Figures 6.1 through 6.3 show the trend of maximum precision and recall as a result of the incremental change in distribution weight, for each heuristic. The max precision lines for edit distance and string matching are almost identical. As string matching accounts for more of the final weight, the recall gradually drops off, whereas for the edit distance it peaks at around 0.4 and then also decreases. For bag of words, there is a sharp decrease in precision when setting its percentage of final weight beyond 0.6, signifying that, while bag of words is impacts the final weight more than either of the two string-oriented heuristics, it should not be assigned too much more importance than the collective results of string matching and edit distance. Given that bag of words is able to achieve perfect precision when assigned a weight of 0.6, it relies on the other two heuristic to perform the ancillary function of reaching a mutual distribution that returns the maximum recall.

Table 6.3: Distribution of weights for heuristics (at threshold 0.7) which maximize precision/recall

Threshold	Weight for Bag of Words	Weight for String Match	Weight for Edit Distance	Precision	Recall
0.7	0.6	0.1	0.3	1.00000	0.96859
0.7	0.6	0	0.4	1.00000	0.96859

Table 6.3 presents the possible weight distributions for string match and edit distance which maximize precision and recall with a bag of words providing 0.6 of the final weight and a threshold of 0.7 on predicate final weight values. As can be seen, either assigning a weight of 0.1 or 0.0 results in, supposedly, no difference between optimal precision and recall. The edit distance heuristic can perform the same type of evaluation as string match (comparing two strings), but requires more overhead. String match was introduced primarily to process obvious cases instantly, without invoking edit distance. It is hypothesized that, in future tests examining the scalability of the implicit linking system, string match will help to cut down on the time it takes to run the program.

6.2.2 Results for Individual and Combined Heuristics

Evaluation of how each heuristic individually compares with the combined heuristic approach taken by the system was also performed, to determine whether each heuristic was actually effective. The literal-entity mapping performed by the Entity Resolver was used as a baseline for precision and recall (it's not considered a heuristic in the same sense as the three listed below are). The differences in performance are shown in Table 6.4, with the top two values being the result when precision is maximized (preferred) and the bottom two when recall is maximized. The results are returned after enumerating all threshold and weight distribution combinations. Bag of words is shown to actually achieve lower than base performance when precision is maximized, which could be because Data.gov predicates can actually be filtered fairly reliably by using the local names as determinants for state-relatedness.

Table 6.4: Max Precision/Recall and Precision/Max Recall optimal pair results for heuristics

Precision and Recall of Individual and Combined Heuristics					
	Literal-Entity (L-E) Mapping	L-E Mapping + Bag of Words	L-E Mapping + String Match	L-E Mapping + Edit Distance	L-E Mapping + All Heuristics
Max Precision	0.779	0.67535	0.99705	0.99706	1.0
Recall	1.0	0.88220	0.88482	0.88743	0.96859
Precision	0.779	0.36004	0.30222	0.30222	0.67851
Max Recall	1.0	1.0	1.0	1.0	1.0

This is shown by the performance of the name-oriented heuristics, string match and edit distance, which improve base performance considerably. The string match and edit distance results are very close, which is not unexpected, considering that the former heuristic returns a value of 1.0 on behalf of edit distance whenever it succeeds. The edit distance has a slightly higher performance, especially in recall, because it goes further than the string matching algorithm to catch the few special cases in which approximate string matching is necessary to identify a predicate name. Combining these two with the bag of words heuristic does achieve perfect precision and a higher recall value than any of the lone heuristics.

7. DISCUSSION AND CONCLUSIONS

Based on the results, it seems that the previously stated hypothesis, (assigning percentage distribution of final weight calculation as bag of words > name-oriented heuristics would yield highest precision), was correct as the optimum precision-recall pair had 60% of the final weight calculation attributed to bag of words. The optimum threshold for predicate final weights was determined to be 0.7, and exploring the maximizing distributions of the remaining string-oriented heuristics (accounting for 40% of final weight calculation) revealed that two possible assignments for β (string match) and γ (edit distance) are <0.1, 0.3> and <0.0, 0.4>. From this it was determined that the string matching heuristic may not be necessary for the functionality of the implicit linking system, as the edit distance appears to be the deciding heuristic. Overall, the final precision and recall values achieved by the evaluation were very promising and stood at Precision = 1.0 and Recall = 0.96859, when preferring accuracy. However, the scalability of the system has yet to be tested, and it is possible that running it on thousands of larger datasets (millions of triples) could yield different results.

One point of interest that was discovered while performing the analysis was the first heuristic, bag of words which looks at the values of the predicates, can sometimes be used to identify erroneous data or new state representations. If the value returned by the heuristic is very close to 1.0, as shown in Figure 7.1 for the *area* predicate, it suggests that the unmatched predicate values may be state-related. Given that *area* is in the gold standard, and therefore a verified state-related predicate, it seems odd that all its values would not mapped to state representations.

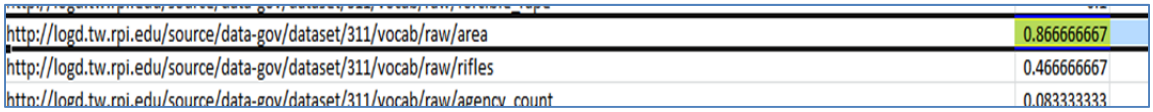


Figure 7.1: Predicate area and certainty measure returned by bag of words heuristic

After performing a query to examine the values attached to *area*, we find that there are misspellings of the names of states, shown in Figure 7.2, which reduce the certainty value returned by bag of words. Thus, while probably not always a definite indicator of erroneous data, the bag of words heuristic can operate to alert the user of problems within the data if the certainty value returned does not meet expectations.

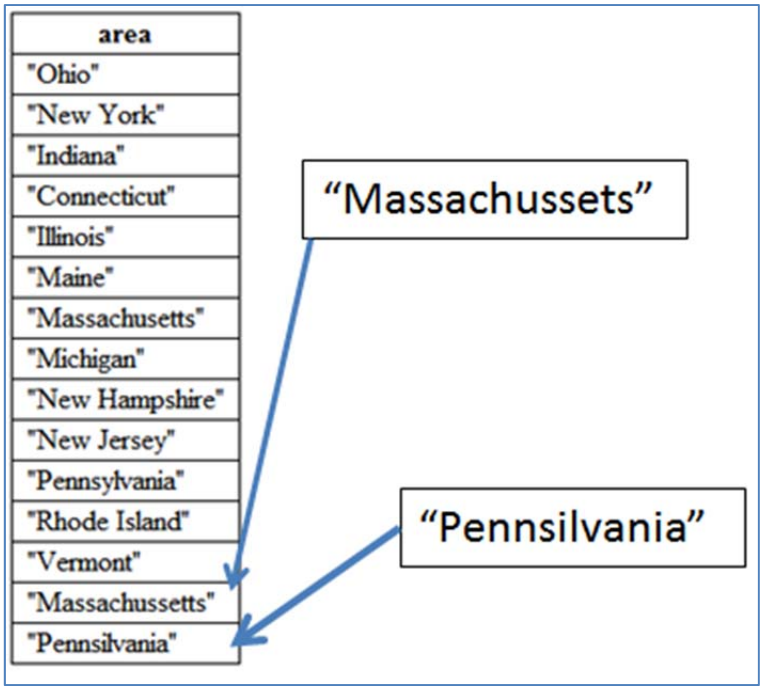


Figure 7.2: Erroneous values attached to area predicate from dataset 311 returned from SPARQL query

A second interesting endeavor explored with the system's output was the visualization of the implicit linking sets. The datasets were dumped into a Simile Exhibit faceted browser to create a demo, shown in Figure 7.3, which allowed filtering over datasets and states, primarily, and some other peripheral metadata. The demo provided the capability (unlocked by implicit linking) for one to draw associations between originally disparate Data.gov datasets by looking for any shared states or territories they possessed. Conversely, the demo, when focused on a single dataset, could display the state-related contents which it contained. Thus, the demo actually becomes more of a tool that users can interact with to play around with ideas for combining related data or even just data discovery.

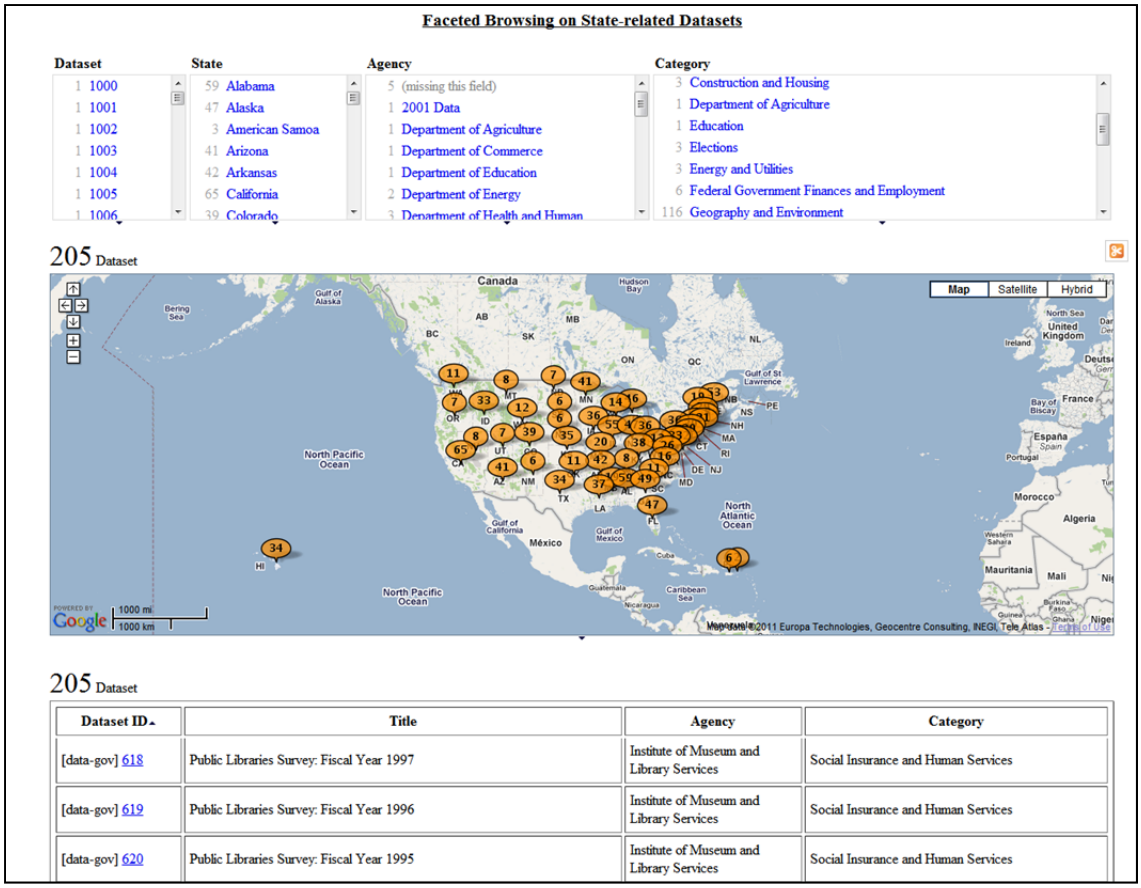


Figure 7.3: Implicit link sets placed in a faceted browser which associates Data.gov datasets with U.S. states and territories

Ultimately, the implicit linking system accomplishes the task of discovering and reporting back instances of U.S. state and territory entities existing within Data.gov datasets. It does this with considerable accuracy and recall, and has the prospect, if expanded, of providing transparency to the data it processes by revealing what interesting entities or categories it contains. The system was written to be as disjoined from the data fed into it and the category it searches for, thereby potentially only requiring the user to provide the required inputs (i.e., mapping set, dataset, search terms) without any type of additional programming or modification to the actual system involved. The inner system logic itself is also exposed by the certainty weights it returns for each heuristic and state-related candidate predicate it encounters. From the three criteria presented in chapter 4, it seems to meet all of them for the most part, thereby presenting itself as a robust tool which could be used to perform unsupervised data linking.

LITERATURE CITED

[1] "Data.gov." Data.gov. United States Federal Government, 2011. Web. Date Last Accessed 10 Apr 2011. <http://www.data.gov/>.

[2] “Linking Open Government Data.” LOGD. Tetherless World Constellation, 2011. Web. Date Last Accessed 17 Apr 2011. < http://logd.tw.rpi.edu>.

[3] Berners-Lee, Tim. “Linked Data”. W3C. W3C, 2009. Web. Date Last Accessed 05 Mar 2011. <http://www.w3.org/DesignIssues/LinkedData.html>.

[4] Rahm, E, and P Bernstein. "A survey of approaches to automatic schema matching." VLDB Journal 10. (2001): 334-350. Web. Date Last Accessed 09 Sep 2010. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.700&rep=rep1&type=pdf>.

[5] Do, H, S Melnik, and E Rahm. "Comparison of schema matching evaluations." In Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems. (2002): 221-237. Web. Date Last Accessed 09 Sep 2010. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.4792&rep=rep1&type=pdf>.

[6] Nottelmann, H, and U Straccia. "sPLMap: A probabilistic approach to schema matching." In 27 th European Conference on Information Retrieval (ECIR ’05). Springer Verlag. 2005. 81-95. Web. Date Last Accessed 10 Sep 2010 <http://www.springerlink.com/content/j15bp7xa1kw8bwrf/fulltext.pdf>.

[7] Shvaiko, P. "A classification of schema-based matching approaches." Journal on Data Semantics 4. (2005): 146-171. Web. Date Last Accessed 09 Sep 2010. <http://www.dit.unitn.it/~p2p/RelatedWork/Matching/JoDS-IV-2005_SurveyMatching-SE.pdf>.

[8] Aumueller, D, H Do, S Massmann, and E Rahm. "Schema and ontology matching with COMA++."SIGMOD '05 (2005): 906-908. Web. Date Last Accessed 09 Sep 2010. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.8145&rep=rep1&type=pdf>.

[9] Bohannon, P, E Elnahrawy, W Fan, and M Flaster. "Putting context into schema matching." VLDB '06 (2006). Web. Date Last Accessed 09 Sep 2010. <http://www.vldb.org/conf/2006/p307-bohannon.pdf>.

[10] Resnik, P. "Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language ."Journal of Artificial Intelligence Research 11. (1999): 95-130. Web. Date Last Accessed 09 Sep 2010. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.3785>.

[11] Sheng, H, H Chen, T Yu, and Y Feng. "Linked data based semantic similarity and data mining." IEEE IRI '10 (2010): 104 - 108 . Web. Date Last Accessed 09 Sep 2010. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05558957>.

[12] Tummarello, G, and R Delbru. "Publishing data that links itself: a conjecture." AAAI '10 (2010). Web. Date Last Accessed 09 Sep 2010. <http://www.aaai.org/ocs/index.php/SSS/SSS10/paper/download/1189/1467>.

[13] Cucerzan, Silviu. "Large-scale named entity disambiguation based on Wikipedia data." Proc. 2007 Joint Conference on EMNLP and CNLL. Prague, Czech Republic, ACL. 2007. 708-16. Web. Date Last Accessed 05 Apr 2011. <http://acl.ldc.upenn.edu/D/D07/D07-1074.pdf>.

[14] Ding, Li and DiFranzo, Dominic and Graves, Alvaro and Michaelis, James R. and Li, Xian and McGuinness, Deborah L. and James A. Hendler"TWC data-gov corpus: incrementally generating linked government data from data.gov." Proceedings of the 19th international conference on World wide web (WWW '10). ACM. 2010. 1383-86. Web. Date Last Accessed 04 Apr 2011. <http://delivery.acm.org/10.1145/1780000/1772937/p1383-ding.pdf?key1=1772937&key2=6772803031&coll=DL&dl=ACM&ip=128.113.195.141&CFID=18250078&CFTOKEN=61962493>.

[15] Pasca, Marius. "Acquisition of categorized named entities for web search." In Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM '04). New York, ACM. 2004. 137-45. Web. Date Last Accessed 04 Apr 2011. <http://delivery.acm.org/10.1145/1040000/1031194/p137-pasca.pdf?key1=1031194&key2=1842803031&coll=DL&dl=ACM&ip=128.113.195.141&CFID=18250078&CFTOKEN=61962493>.

[16] Nadeau, David, and Satoshi Sekine. "A survey of named entity recognition and classification ."Lingvisticae Investigationes. John Benjamins Publishing Company. 2007. 3-26. Web. Date Last Accessed 04 Apr 2011. <http://nlp.cs.nyu.edu/sekine/papers/li07.pdf>.

[17] Ratinov, Lev, and Dan Roth. "Design Challenges and Misconceptions in Named Entity Recognition."Proceedings of the Thirteenth Conference on Computational Natural Language Learning. (CoNLL '09): Stroudsburg, PA, ACL. 2009. 147-55. Web. Date Last Accessed 04 Apr 2011. <http://portal.acm.org/ft_gateway.cfm?id=1596399&type=pdf&CFID=18250078&CFTOKEN=61962493>.

[18] Bizer , C, T Heath, D Ayers, and Y Raimond. "Interlinking open data on the web." ESWC2007(2007). Web. Date Last Accessed 09 Sep 2010. <http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkingOpenData.pdf>.

[19] Raimond, Y, C Sutton, and M Sandler. "Automatic interlinking of music datasets on the semantic web." LDOW'08 (2008). Web. Date Last Accessed 09 Sep 2010. <http://events.linkeddata.org/ldow2008/papers/18-raimond-sutton-automatic-interlinking.pdf>.

[20] Bizer, C, J Lehmann, G Kobilarov, S Auer, and C Becker. "DBpedia - A crystallization point for the Web of Data." Web Semantics: Science, Services and Agents on the World Wide Web 7.3, 2009: 154-165. Web. Date Last Accessed 09 Sep 2010. <http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B758F-4WS9BS0-1&_user=659639&_coverDate=09/30/2009&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&view=c&_searchStrId=1581544490&_rerunOrigin=google&_acct=C000035878&_version=1&_urlVersion=0&_userid=659639&md5=d74704d8dd5f9518872c5887b75ab176&searchtype=a>.

[21] "RDF Primer". W3C. W3C, 2004. Web. Date Last Accessed 10 Apr 2011. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.

[22] "SPARQL Query Language for RDF". W3C. W3C, 2008. Date Last Accessed 10 Apr 2011. <http://www.w3.org/TR/rdf-sparql-query/#introduction>.

[23] "LOD cloud diagram." The Linking Open Data cloud diagram. Web. Date Last Accessed 15 Apr 2011. <http://richard.cyganiak.de/2007/10/lod/>.

[24] "Jena – A Semantic Web Framework for Java." Jena. sourceforge.net. Web. Date Last Accessed 17 Apr 2011. <http://jena.sourceforge.net/>.

[25] "DBpedia". DBpedia. DBpedia. Web. Date Last Accessed 21 Apr 2011. <http://dbpedia.org>.

APPENDICES

A. SPARQL Queries

A.1 Literal-Entity Mapping Example

```
Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix owl: <http://www.w3.org/2002/07/owl#>
Prefix dcterms: <http://purl.org/dc/terms/>
select distinct ?dbpedia_uri ?p ?p_alt ?entity_label
where {
  graph
  <http://logd.tw.rpi.edu/source/data-gov/dataset/353/version/1st-anniversary/conversion/raw/subset/sample>
  {
    ?s2 ?p ?entity_label.
    optional {?s ?p_alt ?o. ?o rdfs:label ?entity_label. }
  }
  graph
  <http://logd.tw.rpi.edu/source/twc-rpi-edu/dataset/instance-hub-us-states-and-territories/version/2011-Apr-09>
  {
    ?s1 dcterms:identifier ?entity_label;
       owl:sameAs ?dbpedia_uri.
  }
}
```

Figure A.1: Example SPARQL query that is used to perform Literal-Entity mapping between Data.gov dataset 353 and a LOGD mapping set

SPARQLer Query Results

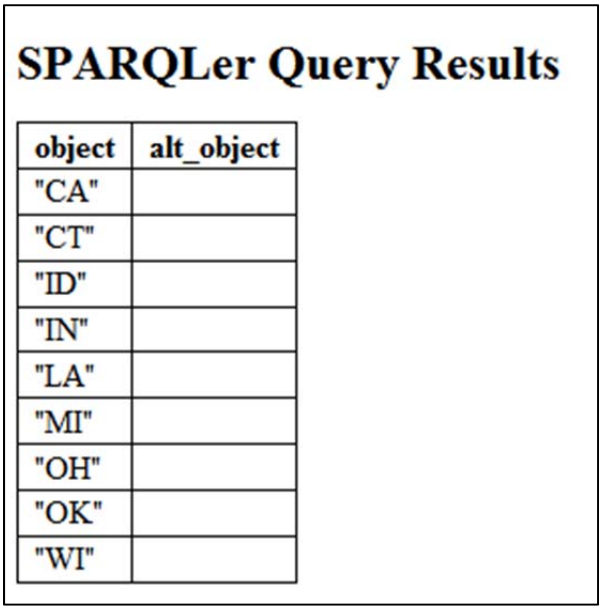
dbpedia_uri	p	p_alt	entity_label
<http://dbpedia.org/resource/Idaho>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/grantaw>		"16"
<http://dbpedia.org/resource/Northern_Mariana_Islands>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/grantaw>		"69"
<http://dbpedia.org/resource/Alabama>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"AL"
<http://dbpedia.org/resource/Alaska>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"AK"
<http://dbpedia.org/resource/Arizona>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"AZ"
<http://dbpedia.org/resource/Arkansas>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"AR"
<http://dbpedia.org/resource/California>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"CA"
<http://dbpedia.org/resource/Colorado>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"CO"
<http://dbpedia.org/resource/Connecticut>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"CT"
<http://dbpedia.org/resource/Delaware>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"DE"
<http://dbpedia.org/resource/Washington,_D.C.>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"DC"
<http://dbpedia.org/resource/Florida>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"FL"
<http://dbpedia.org/resource/Georgia_%28U.S._state%29>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"GA"
<http://dbpedia.org/resource/Hawaii>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"HI"
<http://dbpedia.org/resource/Idaho>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"ID"
<http://dbpedia.org/resource/Illinois>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"IL"
<http://dbpedia.org/resource/Indiana>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"IN"
<http://dbpedia.org/resource/Iowa>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"IA"
<http://dbpedia.org/resource/Kansas>	<http://logd.tw.rpi.edu/source/data-gov/dataset/353/vocab/raw/mail_st>		"KS"

Figure A.2: Results from Literal-Entity mapping example SPARQL query

A.2 Bag of Word Heuristic SPARQL Example

```
PREFIX conversion: <http://purl.org/twc/vocab/conversion/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select distinct ?object ?alt_object
where {
  graph
  <http://logd.tw.rpi.edu/source/data-gov/dataset/249/version/1st-anniversary/conversion/raw/subset/sample>
  {
    ?s <http://logd.tw.rpi.edu/source/data-gov/dataset/249/vocab/raw/facility_state> ?object.
    optional {?object rdfs:label ?alt_object. }
  }
}
```

Figure A.3: Example SPARQL query which finds all objects attached to a predicate for Bag of Words heuristic



SPARQLer Query Results

object	alt_object
"CA"	
"CT"	
"ID"	
"IN"	
"LA"	
"MI"	
"OH"	
"OK"	
"WI"	

Figure A.4: Bag of Words example SPARQL query result

B. Linked Open Data Cloud

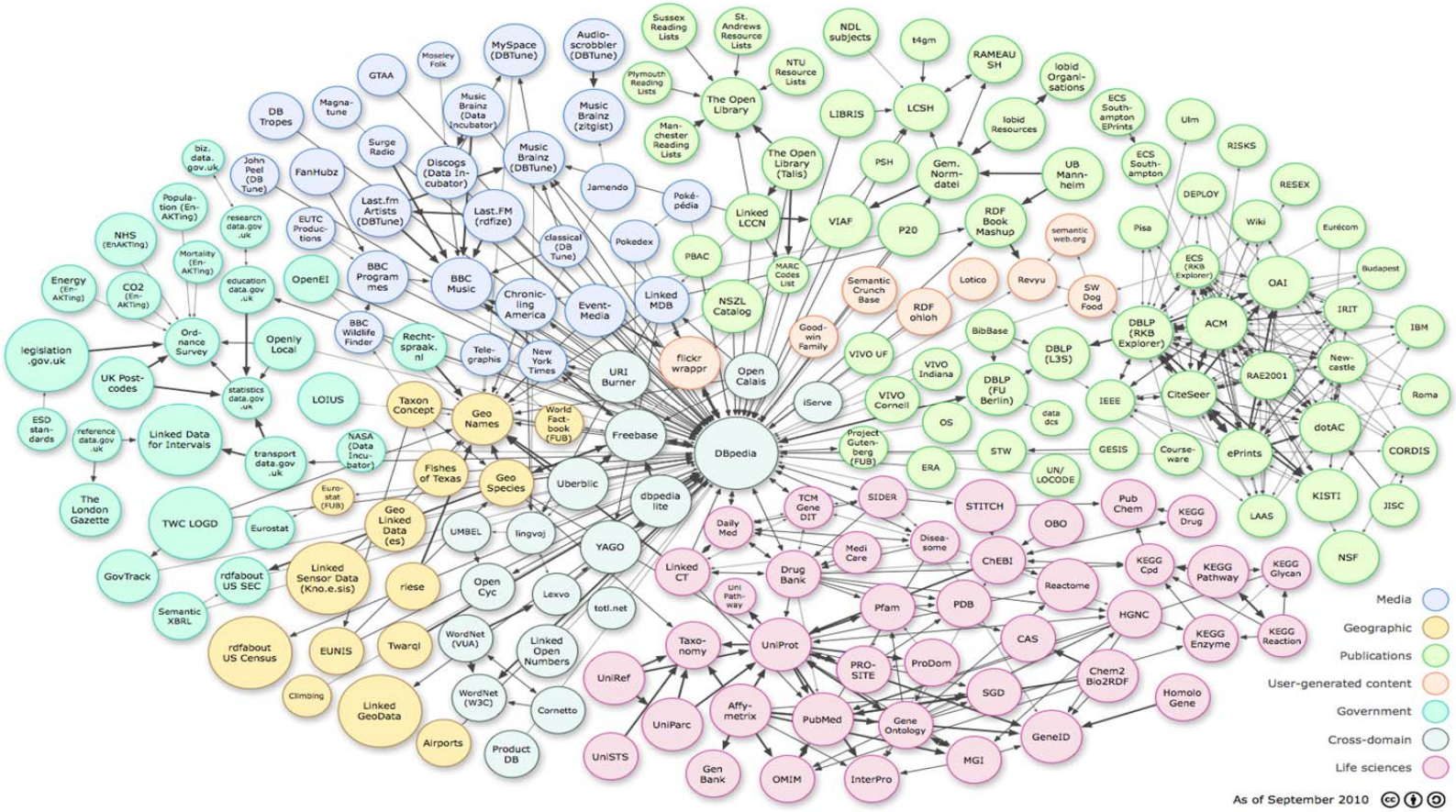


Figure B.1: Linked Open Data Cloud [23], which displays the different datasets published as Linked Data on the web

C. Workflow Diagram

Figure C.1: Workflow Diagram for Automated Implicit Linking System