Semantic Fusion of Live Web Content: System Design and Implementation Experiences

Vincent Lenders
armasuisse, Switzerland
vincent.lenders@armasuisse.ch

***Abstract*—Conventional Web search models are ineffective at providing quick and comprehensive answers to questions related to live content such as real-time data or temporal relationships between actors. Semantic data fusion techniques have the potential to provide a more suitable abstraction model for efficient search on this type of data. However, myriad architectural and technical implementation challenges arise when trying to implement a working system. This paper summarizes our efforts and experiences at implementing a functional semantic fusion system for live content from the Web. Besides semantic data fusion techniques, we make extensive use of natural language processing, semantic Web technologies and Bayesian statistics to render the system a self-contained framework acting directly between Web resources of interest and end-user search applications. We first present the semantic fusion architecture design that we have developed. We have implemented this architecture and tested its effectiveness using real-world live data from the Web over multiple weeks. We then report about our major experiences and lessons-learned of this experiment.**

I. Introduction

The Web offers a tremendous amount of information to its users. To make effective use of this huge information space, online services like Yahoo or Google are maintaining a searchable index of the entire Web. This index can be leveraged by users which issue plain-text queries in order to retrieve a list of ranked Web pages relevant to their queries. This traditional search model has proven to be extremely successful to find static content residing on specific Web sites. However, when searching for live content such as real-time data or temporal relationships between actors that may be dispersed over multiple Web sites, this traditional search model is not well suited. Users have to issue multiple queries one after another, iteratively browse through the page results, and manually infer the answers by reading through the content. This traditional search paradigm is overwhelming and does not provide the appropriate level of abstractions for quick and comprehensive information retrieval.

With the proliferation of the Web 2.0[1], this issue is now even more relevant. Increasingly more live content is available on the Web today representing dynamic data such as geo-referenced objects, sensor data, or social media content. The ability to quickly and effectively search over this Web space without having the user to do most of the information processing manually is thus more needed than ever.

[1]The term Web 2.0 was originally introduced by Tim O'Reilly. The term does not refer to a new technical specification but rather to a natural evolution of the static Web with a more collaborative type of medium with a high-level of user-generated interactions and dynamic content.

A key technology that has the potential to provide a higher level of abstraction for search in live content is *semantic data fusion*. Semantic fusion techniques [1] would allow one to align the data and bring it into a consistent semantic data representation scheme. On top of such a semantic representation model, semantic search services may then be implemented to correlate and reason about the collected information according to a semantic abstraction model that takes away the actual search complexity from the users. A major challenge with this vision is however that live content in the Web is often not structured or the structure of the content is not known in advance. The content must therefore be semantically aligned before semantic data fusion can be applied.

Following a practical research-oriented approach, we have explored different semantic fusion techniques and system architectures to realize semantic alignment and fusion of live content in the Web. The problem is particularly challenging from a systems perspective because besides requiring semantic data fusion techniques, myriad other computer science disciplines that are still being actively researched such as information retrieval, natural language processing, semantic knowledge representation, semantic databases, and Bayesian statistics need to be integrated into a common framework. Our goal was therefore to understand the practical limits of the current state-of-the-art in these fields and to implement a system with open-source software that would come as close as possible to the inherent limitations of today's technologies.

Our major contribution is the design of a semantic fusion system architecture based on semantic web standards that is able to handle uncertain data as found in live Web content. We have implemented our architecture and report about our major experiences with the semantic fusion of real-world data sources.

II. Related Work

Public search services in the Internet have started exploiting semantic data fusion techniques to enhance their results. Most popular examples are WolframAlpha [3] or Google's Knowledge Graph [4] for text-based search and Apple's Siri for voice-based search. While the search abstraction models of these services are similar to what we aim to achieve in this work, the underlying data characteristics differ fundamentally. These services aim at the fusion of systematic knowledge such as expert knowledge derived from sources like the CIA Factbook [5], Wikipedia [6] or Freebase [7]. In contrast to live content, this type of knowledge is rather static, highly certain,

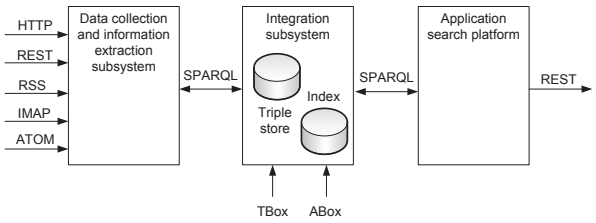

Fig. 1. Semantic fusion system architecture.

and generally available already in structured formats such that the semantic fusion problems we face in this work do not need to be addressed by those systems.

The semantic Web initiative [8] has come up with various models and standards for semantic knowledge representation and reasoning which we adopt in this work. However, the idea of the semantic Web is that content providers publish their contents using the semantic Web standards so that the content is already structured. While individual content providers have started to publish and link their data in accordance to these standards, a majority of the content in the Web, and specially live content, is still provided in an unstructured manner or according to a structure that is unknown.

III. System Design

This section describes our proposed system design to retrieve, store and search live Web content. The architecture consists of three components as shown in Figure 1. The data collection and information extraction subsystem is in charge of retrieving the data form the external Web sources, analyzing their contents and transforming the relevant facts into resource description framework (RDF) triples. These triples are then passed to the integration subsystem over a SPARQL[2] endpoint. At the integration layer, all triples are persisted into a triple store. Indexes of the subjects, predicates and objects are maintained within the triple store. An additional index over all keywords is generated and maintained outside the triple store. The triples are managed in a global context with ontologies represented by the Terminological and Assertional Boxes (TBox and ABox, respectively). TBox and ABox are concepts used in description logics to distinguish between the conception of triples and their particular manifestations. At the last stage, the data is retrieved from the integration layer again through a SPARQL endpoint. Applications issue queries over a representational state transfer application programming interface (REST API) [9] to an intermediate search platform which maps user queries to one or multiple SPARQL queries and determines the confidence of the results.

A. Data Collection and Information Extraction Subsystem

Data retrieval. The data collection and information extraction subsystem connects to content providers in the Web and parses the collected data. Content is either retrieved on a periodic basis over pull protocols or the system subscribes to content providers and waits for the data to pushed to it. Our architecture supports both kinds of schemes with protocols such as the hypertext transfer protocol (HTTP) or REST APIs for pull-based retrieval or protocols such as RSS, ATOM or IMAP for push-based delivery. Appropriate connectors are implemented for the different methods of access.

Information extraction. The content in the Web is often unstructured or it is structured but we don't know its structure in advance. The first step is therefore to analyze the data and extract its meaning. The difficulty of this task depends on the type of input. When the input is structured in the form of e.g. an HTML table, the knowledge is inferred more easily by analyzing the first row of the table which usually refers to the type of data in the columns [10]. However, for entirely unstructured data such as natural text, more sophisticated approaches using natural language processing (NLP) techniques [11] are needed. NLP is the science that aims to derive semantic meaning from human or natural language input. For example, an important NLP technique we extensively rely on is named entity recognition (NER). NER is used to automatically recognize persons, organisations, geo-locations, dates, etc in text. Also the relationship between named entities (e.g., person A is friend with person B) and facts about their actions (e.g., person A was at location Y) is extracted from the text using NLP techniques.

Semantic structuring. A further task of the data collection and information extraction subsystem is to transform the extracted content into a consistent data format for all the different data sources. The data format we use is the one suggested by the resource description framework (RDF). The syntax of RDF is based on the extensible markup language (XML). The core data format is based on the concept of triples which have the form $\{$*subject*, *predicate*, *object*$\}$. For example, one way to represent the notion "The car has the color black" in RDF is as the triple: a subject denoting "the car", a predicate denoting "has the color", and an object denoting "black". As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representations. However, this simple triple data structure may be used to represent any knowledge by forming collections of RDF statements resulting in arbitrary directed graphs.

Handling uncertainty. Two main causes of uncertainties may arise at the data collection and information extraction levels. The content sources in the Web may provide erroneous, contradicting, or outdated data. In addition, knowledge extraction with automated tools like HTML table extraction or NLP are generally error-prone and may therefore lead to misinterpretation of the content. To handle these causes of uncertainties in our system, we introduce a *degree of belief* for recognized entities and facts. The degree of belief can be interpreted as a subjective belief in the sense of a Bayesian probability [12], i.e. a plausibility with values ranging from zero to one. Recognized entities and facts are therefore assigned a degree of belief in the form of additional triples representing the Bayesian probability of the trustworthiness of the subjects, objects, and their relationships. It may be

[2]SPARQL is a RDF query language. The term is a recursive acronym for SPARQL Protocol and RDF Query Language

desired by users to determine later whether a low belief is the result of poor quality of the data source or poor quality of the information extraction. Therefore, we do not represent the subjective belief of the content with a single degree of belief but suggest to keep separate beliefs for different causes of uncertainties.

B. Integration subsystem

Ontologies. The relationships and meaning of the extracted RDF triples are brought into a higher level semantic context with formal ontologies. Inspired by the semantic Web concepts, we model ontologies in the integration subsystem using the ontology web language (OWL). Generic ontologies based on OWL are already available, for example those who define geographical domains [13] or social relationships (e.g., friend of a friend (FOAF)). We rely on these well-defined Web ontologies as far as possible. However whenever necessary, we define domain-specific ontologies that represent things that may only be of interest to our own application context.

The ABox and TBox [14] are used to describe different types of statements in the ontologies. TBox statements describe a system in terms of controlled vocabularies, for example a set of classes and vocabularies. ABox are TBox compliant statements about that vocabulary. TBox statements are sometimes associated with object-oriented classes and ABox statements associated with instances of those classes.

OWL is based on description logics which is a subset of first-order logic that provides sound and decidable reasoning support [15]. An ontology consists of a set of axioms with place constraints on sets of classes and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on data explicitly provided. For example, an ontology describing families might include axioms stating that a "hasFather" property is only present between two individuals when "hasParent" is also present. Therefore, if an individual x can be related to an individual y by "hasFather", then x must be related to y by "hasParent". Besides, there are some other constructors used to relate two properties to impose cardinality constraints on properties. The `owl:inverseOf` prorperty can be used to specify that "hasParent" is the inverse property of "hasChild". When x is related to y with the "hasParent" property, the inverse property can be inferred that y is related to x with the "hasChild" property.

In OWL, any sentence being asserted facts, domain knowledge or reasoning results must be either true or false and nothing in between. Ontologies defined by OWL can therefore not be used for inference and reasoning in the presence of uncertainty as we aim in our work with the asserted degree of beliefs. To overcome this limitation, probabilistic extensions to OWL have been proposed in the literature [16]. However, as these methods have not been standardized and integrated in existing triple stores yet, we do not reason with uncertainty at the level of OWL but we propose to support reasoning with uncertainty at the level of the application search platform.

Triple store. The main task of the integration subsystem is to store and manage the RDF-structured content from the data collection and information extraction subsystem. RDF triples are stored in a repository named a triple store. A triple store is a database system that manages statements for randomized read and write access. Similar to a relational database, information in a triple store is inserted and retrieved via a query language. The query language we adopt is the one that has been defined by the W3C consortium: SPARQL. Over the past years, many RDF triple stores have emerged that support SPARQL and are able to manage up to billions of triples on a single store. While some manufacturer implement native triple stores, approaches based on relational database systems have also been proposed with similar performance [17]. Our architecture is not depending on a particular type of triple store architecture but requires the store to be compliant to the semantic Web standards such as RDF, OWL and SPARQL [18].

We store data in the triple store in an immutable form. Immutable data concepts have become prominent in the context of big data. With immutable data, we don't update (like e.g. an update in the relational database concept) or delete triples, but create a separate record with a timestamp every time an entity's information evolves. This approach has the advantage that we are able to retrieve the entire timeline of events and thus entirely reconstruct temporal data. In addition, erroneous updates due to the underlying uncertainty in the data and its extraction process will not cause older correct values to be erased. Of course, data storage is not infinite, and we need to delete old records at some point. Our current strategy is to purge old records based on timestamps after a configurable sliding time window of interest has elapsed.

The number of triples with an immutable data schema may quickly become huge. It is therefore recommendable to structure the triples into different sets. We use the concept of *named graphs* in RDF which is also available in SPARQL and most RDF stores. Named graphs turn the RDF triple model into a quad model by extending a triple to include an additional context item of the form {*context*, *subject*, *predicate*, *object*}. This model may be used to structure RDF triples that share some commonality. A natural way to define named graphs is for example to define individual graphs for individual days or different Web resources. By providing logical structuring, named graphs may significantly improve the query performance by reducing the number of triples that need to be considered when issuing SPARQL queries with a specified named graph.

SPARQL endpoints. The data collection and information extraction subsystem as well as the application search platform interact with the integration subsystem over SPARQL endpoints. The SPARQL endpoints are exposed over the HTTP protocol to facilitate interoperable remote access at the cost of a higher overhead over native access protocols. In the current SPARQL version 1.1, the query language supports reading, writing, as well as reasoning over the query language. Both the data collection and information extraction subsystem as well as the application search platform perform read and write operations. The former inserts new RDF triples as new data arrives. In addition, existing triples and the stored ontology may need to be queried resulting in read operations. The application search platform primarily reads information from

the triple store, although it generally makes sense to persist relevant query results with new triples as to improve the query performance.

Index. We further maintain an index of all the keywords from the extracted content. This index provides scalable keyword search in addition to the more powerful semantic search over SPARQL. This is useful to complement semantic search and to provide access to the raw unprocessed text information when necessary.

C. Application search platform

The application search platform acts as a gateway between search applications and the integration layer. This additional subsystem is designed to (i) map canonical user queries to SPARQL, to (ii) infer the degree of belief of replies, and to (iii) provide access to the raw unprocessed information as retrieved from the Web.

Query mapping. The application search platform exposes a REST API to user applications. A REST API is an architectural style that abstracts the architectural elements within a distributed hypermedia system [9]. Upon user application-initiated search requests, queries are mapped to one or multiple SPARQL queries in order to retrieve the response from the integration layer. This level of indirection over a REST API strongly simplifies the design of user applications by abstracting the semantic structure of the underlying RDF-based data modelled as ontologies. The user application is therefore generally not aware of the defined system ontologies and semantic standards. The application search platform further allows to map complex user requests to multiple SPARQL queries, reducing again the complexity of user applications.

To optimize the overall query performance, the application search platform further persists inferred relationships and facts in the triple store with new triples. This is done either for interesting knowledge that has a high level of certainty or for inferred knowledge resulting from SPARQL queries which are frequently issued or have a high computational overhead due to their complexities.

Belief propagation. A core task of the application search platform is to derive the degrees of belief of user requests. Reasoning with OWL within the triple store is limited by first-order logic and therefore cannot cope with uncertainty. Belief propagation according to the degree of belief for concepts identified by the data collection and information extraction subsystem is therefore handled here. The core idea is to construct Bayesian networks as directed acyclic graphs based on the causality from the ontologies or data fusion topologies. We apply Bayesian network inference algorithms such as [19] to propagate the belief of facts to fused knowledge. To propagate the notion of different types of uncertainties, separate Bayesian networks are formulated for different types of uncertainties. For an overall estimation of the belief, joint networks are used.

Raw data retrieval. In addition to replying to semantic user requests over SPARQL, our system provides direct access to the raw content that has been retrieved from the Web. This is for example useful when the degree of belief is low such that the end-users may want to verify query replies manually by going over the raw information. In addition users may on occasion want to see additional information from the raw text which has not been extracted automatically by the data collection and information extraction subsystem. This additional raw information is pulled by applications over the REST API. The application search platform itself retrieves the raw information from the integration subsystem which maintains references to raw content in triples or in the index.

IV. Semantic Data Fusion

Our system is designed to perform semantic data fusion at various levels. This section highlights the places where fusion occurs and how it is implemented.

Text fusion. The first type of semantic fusion is at the data collection and information extraction subsystem level. Each connector retrieves content from specific Web resources. In order to identify objects and their semantic relations from these sources, it is necessary to fuse statements and assertions from different sentences, tables, or paragraphs. This type of fusion is thus the main task of the NLP engine. The fusion results are here the produced RDF triples of the data collection and information extraction subsystem.

Fusion of ontologies. Another place where semantic fusion is required is at the level of ontologies in the integration subsystem. As mentioned earlier, our goal is to use well-known ontologies as much as possible and to introduce our own ontologies only for the modelling of specific domain knowledge. When multiple ontologies are used simultaneously, a mapping between the ontologies is needed to define the concept representations of the different ontologies that refer to the same things. We use two different techniques for ontology fusion. The first technique is the use of the `owl:equivalentClass` property in OWL which links two concepts together by indicating that the two concepts from the different ontologies actually refer to the same thing. For example, we state that the concept for persons in FOAF refers to the same concept for persons that we use in our own ontology by stating `foaf:Person owl:equivalentClass own:Person`. This property is symmetric. The second method consists of using the `rdfs:subClassOf` property of the RDF schema (RDFS). In contrast to the equivalence property in OWL, this property defines an asymmetric relationship. This could for example be used to link the concept of persons in FOAF with our own class for man as `own:man rdfs:subClassOf FOAF:Person`. By linking different concepts from the different ontologies together, we obtain a fused representation of the world.

Multi-source data fusion. Finally, multi-source data fusion combines information extracted from different Web resources. This happens both at the integration subsystem and at the application search platform. At the integration subsystem, RDF triples originating from different Web sites are combined together in the triple store. Since we cannot assume that a single individual will have the same name on different sources, we need to semantically fuse different entities when we realize that they refer to the same individuals. This can be inferred implicitly at the integration subsystem in the triple store using SPARQL queries or explicitly in the application

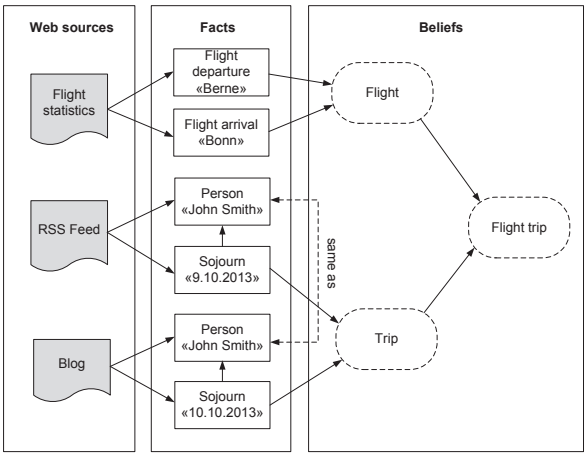

Fig. 2. Semantic data fusion example. A flight trip is inferred from live air traffic information and locomotion information extracted from live reports in a RSS feed and a blog.

search platform. In both cases, the `owl:sameas` property is used to fuse and persist the entities that refer to the same individuals.

Example. Figure 2 shows an example of semantic fusion. The diagram is logically divided into Web sources, facts and beliefs. The example represents a case where live air traffic information is fused with live reports from persons to identify flights that passengers may haven taken to move from one location to another. Continuously updated flight schedules coming from a Web site are used to infer facts about flights from departure and arrival times. RSS feeds and blogs are used to extract information from persons such as names and possible sojourns. Facts from an RSS feed and a blog about a person named John Smith with sojourns on 9.10.2013 and 10.10.2013 have been identified from NLP text analysis. For simplicity, we declare in this example that identified persons with the same name correspond to the same physical person. So, a `owl:sameAs` relationship is added between the two identified persons to fuse their corresponding profiles. Using the fused information, a belief in a trip is inferred based on the two sojourns which have close dates. Finally, by fusing information from the inferred flight and trip, a belief in a potential flight trip is inferred for John Smith from Berne to Bonn.

V. Implementation

To gain practical experience with our system architecture, we have implemented a prototype system in Java and tested it with real input data. We rely as much as possible on existing software frameworks and tools to implement standard modules of the different subsystems. There are currently various open-source and commercial software product alternatives that may be used to support the realization of the target system. We have experimented with some of those to identify the most suitable ones. Our current implementation is based on the following tools. The data collection and information extraction subsystem relies on Apache Camel [20]. Apache Camel is an open-source framework based on known enterprise integration patterns that allows you to define routing and mediation rules in a variety of different languages. For natural language processing, we use GATE [21] and its information extractor ANNIE. At the integration layer, we rely on the Bigdata triple store [22]. Bigdata is an open-source triple store that scales horizontally by allowing to distribute the store on a cluster of nodes. We implement SPARQL 1.1 end-points over a light weight REST API provided by the NanoSparqlServer. Elasticsearch [23], a search server based on Apache Lucene [24] is used to provide a searchable index of the encountered keywords. Search applications are written with Eclipse RAP in Java and exposed as Web services.

We have integrated different types of Web connectors for live content coming from HTML pages, RSS feeds and Email newsletters. The data collection and information extraction subsystem has been operational with 6 connectors for several weeks collecting around 5.7 GB of dynamic Web content per day. The pull-based connectors retrieve new content from the external Web sites once per hour. This data resulted in roughly 1 Million of RDF triples per day, corresponding to 500 MB per day of data in the triple store. Three generic search applications have been implemented providing semantic, keyword and visual search interfaces to end-users. We ran the complete system on a virtualized Linux server with 8 cores and 16 GB of RAM.

VI. Experiences and Lessons-learned

In this section, we describe our major system design and implementation experiences.

Natural language processing errors. Despite many years of extensive research in natural language processing, it still remains a challenge to accurately interpret text semantically with a machine. Untrained commercial and open-source out-of-the-box tools [21], [25], [26], [27] have serious difficulties to analyze texts that differ linguistically from the default training sets provided by their distributors. In our experience with different NER tools on English live reports, we had for example modest precision and recall rates ranging between values from 0.7 and 0.9 for the recognition of names, locations, organisations, dates, etc. These values were then significantly worse and sometimes dropping below 0.5 when considering other languages or more informal texts with slang or (intended) misspellings.

Fusion of ontologies. Combining general and domain specific ontologies together is a huge effort when done manually: it requires an expert to identify identical or related concepts from the different ontologies and link them together with the `owl:equivalentClass` and `rdfs:subClassOf` properties, respectively. To alleviate this problem, several approaches could be used [18]. The most basic methods try to exploit the linguistic labels attached to the concepts in source and target ontology in order to identify potential matches. Other methods use statistical, structural or logical techniques to determine correspondences between concepts. While some tools such as the R2R framework [28], Limes [29] or sameas.org [30] are available, it still remains a challenging

research problem to automatically create correct mappings in practice [31].

Modelling uncertainty. Uncertainty arising from contradictory, missing or misinterpreted information is a common issue when integrating real-word data from different Web sites. The basic ontology language OWL relies on first-order logic and is hence not able to cope natively with uncertainty. We have introduced a degree of belief for triples as a Bayesian probability that propagates trough a Bayesian network in order to express the degree of belief of query results. However in general, determining a precise Bayesian probability for a fact or thing is not a trivial task because not enough a-priori data is available to determine an a-priori probability distribution. A solution that we are considering as future work is to use imprecise probabilities that specify a range of probabilities instead of exact probabilities. The belief propagation could then be modelled using Dempster-Shafer theory [32] or Credal networks [33] instead of Bayesian networks.

Software frameworks. One challenge we encountered during our implementation work was the selection of appropriate software frameworks to build our system. There are currently various open-source and commercial software product alternatives for NLP, NER, triple stores or indexing that may be used to support the realization of the target system. However, despite the abundance of existing tools, it remains a challenge to implement a stable system environment as many of these tools are still under active development and require active testing to identify their strengths and weaknesses. As the field is still relatively new, new tools frequently arise and existing software projects get abandoned. Also, coupling and pipelining tools from different sources is generally a difficult task as the development is uncoordinated and version updates of one tool may lead to incompatibilities with the rest of the system.

VII. Conclusions

We have proposed a system architecture for semantic data fusion of live data coming from the Web. Our system architecture combines different computer science disciplines including information retrieval, natural language processing, knowledge modelling with uncertainties and semantic data fusion. Our prototype implementation has shown the feasibility of the system design. Yet, different practical challenges have been identified. Our goal is to address these open challenges in future work.

VIII. Acknowledgements

We are specially grateful to Tobias Hofer and his team from basis06 for their continuous support in the different implementation phases of this project. We would also like to thank Albert Blarer, Marc Lieber and Andreas Mengel from Trivadis for the fruitful discussions and valuable feedback on the topics of semantic Web technologies and natural language processing. Special thanks also to Philippe Luginbühl from armasuisse for his feedback.

References

[1] H. B. Mitchell, *Data Fusion: Concepts and Ideas*, 2nd ed. Springer Verlag Berling Heidelberg, 2012.

[2] D. S. Alberts and R. E. Hayes, "Code of Best Practice for Experimentation," in *CCRP publication series*, July 2002.

[3] WolframAlpha. (2013, August) Wolfram Alpha search Engine. [Online]. Available: http://www.wolframalpha.com/

[4] Google. (2012, May) Introducing the Knowledge Graph: things, not strings. [Online]. Available: http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html

[5] CIA. (2013, August) The World Factbook. [Online]. Available: https://www.cia.gov/library/publications/the-world-factbook/index.html

[6] (2013, August) Wikipedia. [Online]. Available: http://www.wikipedia.org

[7] (2013, August) Freebase. [Online]. Available: http://www.freebase.com

[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001.

[9] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[10] D. W. Embley, C. Tao, and S. W. Liddle, "Automating the extraction of data from html tables with unknown structure," *Data Knowl. Eng.*, vol. 54, no. 1, pp. 3–28, Jul. 2005.

[11] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press, 1999.

[12] B. de Finetti, *Theory of probability*. New York, NY, USA: J. Wiley & Sons, 1974.

[13] W3C. (2013, August) Geospatial Ontologies. [Online]. Available: http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

[14] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[15] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The description logic handbook: theory, implementation, and applications*. New York, NY, USA: Cambridge University Press, 2003.

[16] Z. Ding and Y. Peng, "A Probabilistic Extension to Ontology Language OWL," in *Annual Hawaii International Conference on System Sciences (HICSS)*, Hawaii, January 2004.

[17] C. Bizer and A. Schultz, "Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints," in *International Workshop on Scalable Semantic Web knowledge Base Systems (SSWS)*, October 2008.

[18] G. Antoniou, P. Groth, F. van Harmelen, and R. Hoekstra, *A Semantic Web Primer*. Cambridge, MA, USA: The MIT Press, 2012.

[19] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, Sep. 1986.

[20] (2013, August) Apache Camel. [Online]. Available: http://camel.apache.org

[21] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A framework and graphical development environment for robust NLP tools and applications," in *Proceedings of the 40th Annual Meeting of the ACL*, 2002.

[22] Systap. (2013, August) Bigdata. [Online]. Available: http://www.systap.com/bigdata.htm

[23] (2013, August) Elasticsearch. [Online]. Available: http://www.elasticsearch.org/

[24] (2013, August) Apache Lucene. [Online]. Available: http://lucene.apache.org/core/

[25] BasisTechnology. (2013, August) Rosette Linguistics Platform. [Online]. Available: http://www.basistech.com/text-analytics/rosette/

[26] (2013, August) Stanford Natural language processing. [Online]. Available: http://nlp.stanford.edu/software/index.shtml

[27] (2013, August) Apache OpenNLP. [Online]. Available: http://opennlp.apache.org/

[28] C. Bizer and A. Schultz, "The R2R Framework: Publishing and Discovering Mappings on the Web," in *1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai*, 2010.

[29] A.-C. Ngonga Ngomo and S. Auer, "LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data," in *Proceedings of IJCAI*, 2011.

[30] sameAs. (2013, August) interlinking the Web of Data. [Online]. Available: http://sameas.org/

[31] (2013, August) Ontology Alignment Evaluation Initiative. [Online]. Available: http://oaei.ontologymatching.org/

[32] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ, USA: Princeton University Press, 1976.

[33] F. G. Cozman, "Credal Networks," *Artificial Intelligence*, vol. 120, pp. 199–233, July 2000.