Case Study: Book Abacus, Book Recommendations Using Hypertable and the Linked Open Data Cloud

Azhar Jassal

Book Abacus, United Kingdom – website: www.bookabacus.com

KEYWORDS: Hypertable, BigTable, Apache HBase, Apache Hadoop, Apache Nutch, Apache Jena, RDF, DBpedia, Wikipedia, Semantic Web, web crawling, data mining

ABSTRACT: Book Abacus is a data science project that crawls the web to discover readily available books, those that can be purchased. These books are presented to users via recommendation pages that are generated using human-interests gathered from the Linked Open Data cloud. For its data repository, it uses Hypertable, a BigTable column-oriented store.

Discovering books on the web may provide the best estimates on how many actually exist. In 2010 Google used their index of the web to discover almost 130 million books (Taycher, 2010).

Book Abacus discovers in the same manner, by crawling the web. We are primarily interested in gathering data about books that that are available to purchase, this is the body of literature that is readily accessible to billions of people. We believe this data set, of the books that are available for purchase on the web has usages aside from price comparison and market intelligence. An example is data science exercises exploring the vastness of the book space and the body of readily accessible literature.

The Book Abacus website provides availability and pricing information for over 13.5 million books, aggregated from 50 million offers discovered on the web from both physical and digital booksellers around the world. This data indicates which books are accessible to particular regions of the world at a given time. Books are indexed using 4 million detailed human interests (topic recommendation pages). The home page displays a selection of books for topics that appear in UK and US news stories, updating every half an hour. Around 1 million price updates occur daily, fed by our crawling activities (these statistics are as of January 2015).

Figure 1. Recommendation page for a topic, in this instance for "Apollo 11"

To construct the topic recommendation pages (fig. 1), Book Abacus must (1) discover a large number of books that are available to purchase and (2) discover a large number of human interests.

To gather book information, we use an Apache Nutch-based crawler (bot). The behavior of our bot is directed using domain definitions. Based upon Apache Hadoop, the crawl cycle runs as a set of MapReduce jobs in the cloud. Where a set of data items have been discovered that meet our perception of what a book is and business criteria is met, such as purchasing availability, the data items are recorded with provenance information. Using Apache Nutch in this manner provides us with a highly scalable data mining capability. The data collected by bot nodes is shipped to Hypertable where it is aggregated and enriched using domain models to present "book" instances that are consumed by downstream systems.

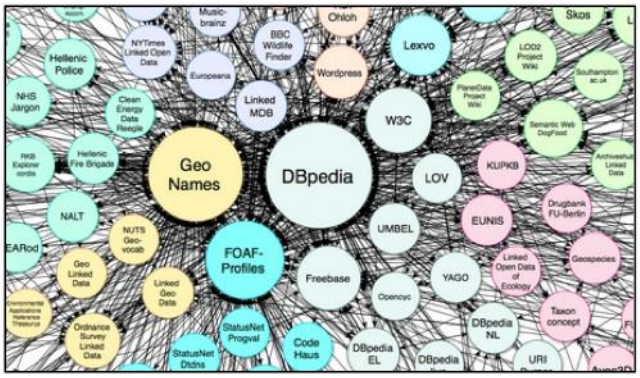


Figure 2. Extract of the Linked Open Data Cloud (Schmachtenberg, Bizer, & Jentzsch, 2014)

To gather human interests, Book Abacus consumes data from the LOD cloud (fig. 2). Linked data is published as RDF. Datasets are typically downloadable en-masse (data snapshots). The LOD cloud contains data sets from government, non-profit and commercial organizations. The LOD cloud is a rich source of human interests. The center point of the LOD cloud is DBpedia, a structured extraction of Wikipedia, downloadable as a RDF data set.

DBpedia contains structured descriptions of conceptual "things" that are generated from Wikipedia articles. As articles are generally entity-centric (one article is about one conceptual "thing"), DBpedia presents a rich, multi-language source of human interests that are used to seed our topic recommendations. Due to its source DBpedia is indirectly human curated. Beyond the descriptions, the link graph within DBpedia captures a crowd sourced human perspective on topic relationships. We analyze this link graph to produce "related" book recommendations. This semantic analysis enables recommendations in the long tail, where purchasing data may not enable heuristic recommendation models (e.g. "people who viewed this also viewed", "people who bought this also bought").

Consumers of data sets from the LOD cloud typically use native RDF databases (triplestores) and SPARQL (a query language for RDF) to persist and interact with the underlying data. Book Abacus uses the BigTable store Hypertable, its query language HQL and MapReduce. It is the central data repository of Book Abacus. Outside of Hypertable we use a variety of natural language processing (NLP) and search technologies to enrich book data and produce our topic recommendation pages.

BOOK ABACUS AND HYPERTABLE

In order to determine the essential features Book Abacus required from a column-oriented store, we undertook a 6-month period of analysis, studying the data our bot was collecting and the data gathered from the LOD cloud. In order to maximize data value it became apparent that the ability to query using orthogonal properties was vital to exploiting relationships. This made secondary indexes an essential feature. Using secondary indexes, data can be arranged appropriately at load-time to enable fast query-time performance when selecting from properties other than row (primary) key. The advantage of being able to specify secondary indexes on columns is that additional cost is only incurred for the relationships our use-cases exploit. This is important in the context of persisting and querying a large amount of RDF where every fact presents a relationship.

Secondary indexes enable fine-grained control over the RDF properties that are indexed and those that are just stored, in similar fashion to how a field in an Apache Lucene document can be set as stored, indexed or both. Triplestores generally index each property in the same manner, which results in low-value properties bearing the same cost as high-value properties. While treating properties equivalently facilitates discovery, significant cost can be unnecessarily incurred due to under utilization. We prefer a less-is-more approach, where the cost of every data item is attributable. For example, if we discover a property joining a person with a birthplace and determine that there is a use-case for that relationship, we can set a secondary index on that joining property and build a query to exploit it. This gives us significant control over the data we are managing and the subsequent cost of its utilization in delivering use-cases.

Hypertable is a high performance column-oriented store. This case study discusses how we use (1) secondary indexes for querying via orthogonal properties and data integration and (2) atomic counters for counting relationships within RDF data sets and subsequent topic ranking.

Storing a RDF triple in a column-oriented store is straightforward. There is a one-to-one mapping between a triple and a column. Triples are made of three parts, the subject, predicate and object. Columns are composed of a row key, the column, column qualifier, cell value and last modified time. With basic mapping, a single column can accommodate a RDF triple. The subject of the triple becomes the row key. The cell value is the object. Depending upon the value of the relationship, we determine how it is to be exploited and select an appropriate column from the table schema to match our usage, e.g. using a store-only or value secondary-index column. The predicate itself is stored as the column qualifier. Multiple column types can exist in the schema. These are reused, for example: columns that only store data (used for low-value properties) and those with secondary-indexes (used for high-value properties). The last modified time is set to present time, but can be overridden. Last modified times are used for data provenance, e.g. recording when the data item was seen by our bot. This allows us to calculate "last seen" times per individual data item.

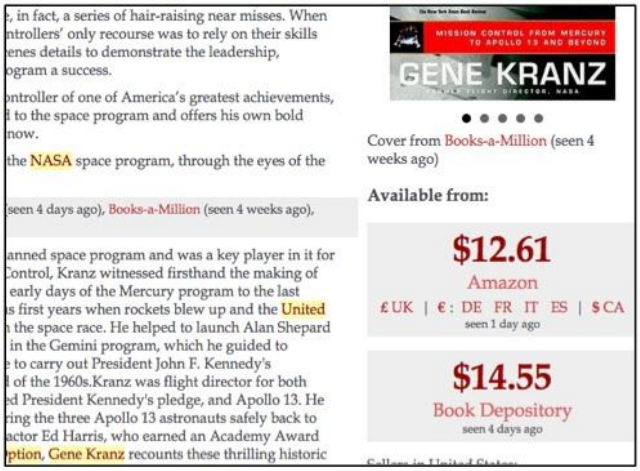


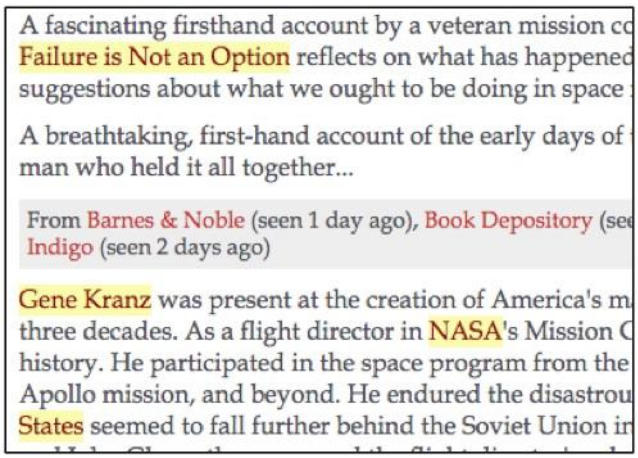
Figure 3. Focused book details, for "Failure Is Not An Option" by Gene Kranz, purchasing information on the right

Hypertable provides native secondary indexes (introduced in 0.9.5.6, improved significantly in 0.9.8.0) on both column qualifiers and cell values. Querying data using orthogonal properties is a difficult problem for column-oriented stores. Numerous implementations exist to provide secondary indexes within HBase, but each has limitations and drawbacks. Hypertable provides a clean native implementation that can be used out-of-the-box.

We utilize secondary indexes to quickly integrate book data at load-time from a large number of continuously changing sources. As the data has been arranged appropriately at load-time, we are able to retrieve an integrated model of a book for book detail pages (fig. 3) with minimum computational cost at query-time. The book detail pages offer purchasing information and compound book details to provide multi-source attribution.

Using standard book industry identifiers and querying orthogonal properties, we are able to inexpensively integrate book data using Hypertable. We retrieve the integrated model from Hypertable and instantiate a weighted provenance model of a book instance. The model is constructed using similarity and duplicate content analysis, where every individual data item per source is a vote for its correctness. This allows us to discover the most used data properties that describe a book, and the rare, unique and false cases. Additionally, we are able to influence the weighting factors for individual sources in order to make sources more or less authoritative.

Every data item in the weighted provenance model is annotated with data that includes the source URL of where it was found and the time seen. We can therefore attribute each individual data item up to the view layers of our consuming applications. Business rules can also be implemented to enable dynamic weighting. An example of dynamic weighting is influencing data from booksellers in the same country of publication as a given title (preferring Swedish book data sources for a Swedish book).


A fascinating firsthand account by a veteran mission cc
Failure is Not an Option reflects on what has happened
suggestions about what we ought to be doing in space

A breathtaking, first-hand account of the early days of
man who held it all together...

From Barnes & Noble (seen 1 day ago), Book Depository (see
Indigo (seen 2 days ago)

Gene Kranz was present at the creation of America's m
three decades. As a flight director in NASA's Mission C
history. He participated in the space program from the
Apollo mission, and beyond. He endured the disastrou
States seemed to fall further behind the Soviet Union in

Figure 4. Focused book details, showing book data compounding and enrichment (topic highlights)

The book detail pages display data items ranked by perceived correctness (fig. 4). Below each data item, attribution is placed providing a link to where the data was seen and how long ago from present time. Descriptions are enriched using NLP and entity extraction (note highlights in fig. 4). The corpus for the entity extractors are the human interests gathered from the LOD cloud. In the example shown in fig. 4, the associated topics are: Failure Is Not An Option, NASA, Gene Kranz and United States. When a user clicks on an inline highlight, they are taken to the corresponding topic recommendation page. This enables a user to traverse the human-interest link graph through book descriptions. The recommendations are scored using a relevance algorithm for which a key factor is incoming and outgoing link count. Hypertable facilitates this by providing an atomic counter column type.

Atomic counters are incremented, decremented or reset by inserting a value with a positive, negative or zero integer. As data is loaded into Hypertable, values are written to counter columns. Maintaining counts at load-time avoids expensive IO-bound scans that have to comb through large amounts of data. The logic to increment or decrement counters exists within the loader. For highly interlinked RDF data sets, we find it valuable to count the number of incoming and outgoing links to and from each resource. These counts and others are factored in producing related topic recommendations.

Figure 5. Focused thumbnail grid showing related topics for George H. W. Bush

Thumbnail grids (fig. 5) appear on topic recommendation and book detail pages to present related topics ranked using a score that factors counts held within Hypertable. Using JavaScript and AJAX calls, the web browser asynchronously fetches related thumbnails. The response from the server includes a HTTP header containing the related score of that topic. The thumbnail and associated link is then inserted into the grid, with the position selected by a score comparator that reads scores from the response headers. Related topic scores also influence the contents of the homepage, which is regenerated every half an hour. It contains topics that are deemed to be popular at that time (popular topics) and a selection of recommended books for each.

To identify popular topics we gather news stories from UK and US sources. Each news story is processed using the same NLP and entity extraction components that are used to enrich book descriptions with inline highlighting

(discussed earlier). The resulting associated topics from current news stories are deemed popular topics. An algorithm is used to select books for those topics that factors various counts. These include the number of present offers and even property specific criteria, e.g. a count of known cover thumbnails URL's. These counts are maintained in Hypertable counter columns and influence the selection of books that appear on the home page. Every story is given an aggregated topic score, for example a story about a US election may have the associated topics: 'Barack Obama' and 'United States'.

Stories and books about such topics are likely to appear towards the top of the home page, as the aggregated topic score is likely to be significant. Such scoring provides interesting metrics. For example, uncovering opportunities for authors by finding topics for which few books are presently available or were ever written. Such data science exercises may provide interesting guidance to authors and publishers looking to discover present gaps and opportunity areas. We plan to undertake such data science exercises in the coming future.

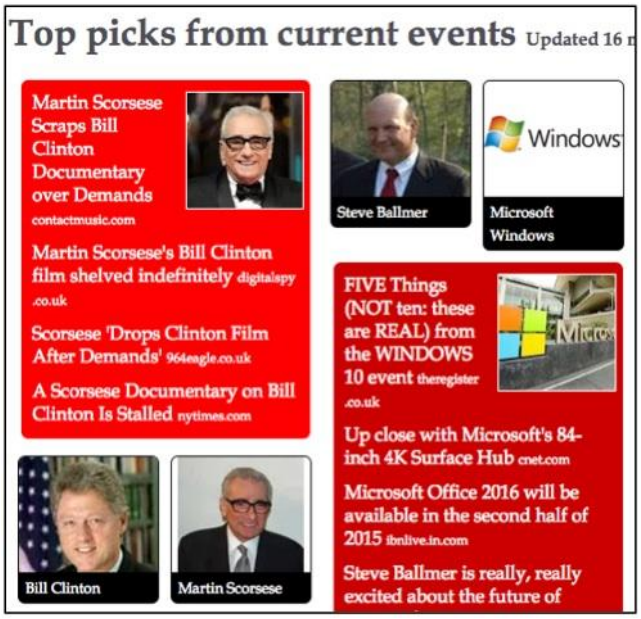


Figure 6. Home page, showing news stories and associated topic thumbnails that link to recommendation pages

The home page is designed to be highly reactive to world events. It will display up to 40 news stories, associated topic thumbnails and 6 recommended books for each. The number of news stories and books displayed to an end-user depends upon device screen size. The home page is responsive; users accessing the site from mobile devices will be presented fewer recommended books per news story in order to display clear relationships and maintain user-experience. The books recommended for each topic is based on numerous factors that include relevance, purchasing availability and even the number of known thumbnail covers. Filters exclude books that may be inappropriate for all-ages; books that seem too similar when summarized and duplicate suggestions. Some book suggestions that appeared on the home page in January 2015 include those for topics: Ebola Virus, Prince Andrew, North Korea/ Sony, Elon Musk/ SpaceX, Li Ka-shing, Bill Clinton, Tom Brady and King Abdullah of Saudi Arabia.

FUTURE WORK

We continue to identify high-value data science exercises that can be undertaken in the short-to-medium term by actively studying the data gathered by our bot. In the immediate future we plan to increase our crawling capacity, improve the methods by which we generate recommendations and deliver subject recommendation pages (to complement the topic based). For example, topic recommendation pages for the boxers "Mike Tyson" and "Floyd Mayweather" exist, but using DBpedia, YAGO and OpenCyc we can create high-level subject pages that offer books about "American Boxers who won world titles". These pages will also interestingly interlink topic pages.

We will scale our data handling capacity in order to facilitate increasing data exercises. This will involve scaling our Hypertable cluster. Apache Jena, an RDF and semantic web Java library used by Book Abacus to stream RDF to and from Hypertable is to incorporate support for the utilization of Hadoop for RDF processing in an upcoming release (patch submitted by Cray Inc.). We plan to use that functionality within Apache Jena to improve our preprocessing of data sets gathered from the LOD cloud, running jobs before data is imported into Hypertable.

AUTHOR INFORMATION

Azhar (Az) Jassal is a London, UK-based data engineer. Prior to Book Abacus, he worked for: JPMorgan (corporate and investment bank), Metropolitan Police (Digital Policing), the science journal Nature and the Book Depository (now an Amazon company).

ABBREVIATIONS

RDF resource description framework. LOD linked open data. HQL hypertable query language. NLP natural language processing.

REFERENCES

Schmachtenberg, M., Bizer, C., & Jentzsch, A. (2014, August 30). *Linking Open Data cloud diagram 2014*. Retrieved January 23, 2015, from Linking Open Data cloud: http://lod-cloud.net/

Taycher, L. (2010, August 5). *Books of the world, stand up and be counted! All 129,864,880 of you.* Retrieved January 23, 2015, from Google Books Search: http://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html