1-6-06 Notes
Update 1-13-06
Glenn Takanishi
Neuralmachines

Metadata Based Text Search

Overview

The problem is one of querying a large text data corpus for information. The query language should be expressive enough to build relatively complex concepts and accurately retrieve the correct answers.

We represent the textual data elements using RDF/N-triples [1] which are elementary concept atoms composed of subject-predicate-object, SPO, words. These link elements are interconnected nodes in a graph. The subject and objects are the nodes, and the predicate or verbs are the links in the graph.

These SPO triplets have the form analogous to the 3-tuples used in classical KL-One type of programs.[1]
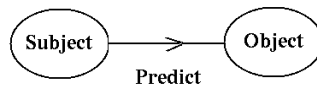

Figure 1: **RDF Triple**

An RDF graph is composed of a set of RDF triples. The direction of the flow of action from subject to object makes the RDF triples in graph language terminology a directed graph.
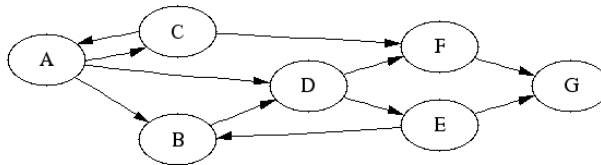

Figure 2: **Directed RDF Graph**

When the number of RDF nodes becomes large as in the representation of a textual document, then the computer resources required to perform pattern matching increases exponentially as a function of the number of nodes, n. With n easily approaching a million nodes for a set of documents, the exact solution to the pattern matching problem becomes intractable.[2]

---

1 KIF, Knowledge Interchange Format Reference Manual, Version 3, Genesereth and Fikes, June 1992

2 Metadata is data describing the data and itself. Extracting meaning from data requires the mechanical data extractor to assume a proper context for the data. Context is acquired by semantic pattern recognition methods such as verb (predicate) ordering, and entity (subject) classifcation. As much as possible, context can be acquired indirectly from a hierarchical ontology used as the pattern matching template. Supervisory training on selected sample datasets is

Theoretically, the complexity to the pattern matching problem increases exponentially as a function of the number of documents. But we avoid this problem by limiting the context of the documents we are reading, and assigning RDF triples to. Limiting or constraining document context is done by having our RDF metadata extractor use a specific set of SPO words.

RDF records are created by reading the text data using the Brill[3] parts of speech, POS, tagger. This mechanical extractor called **meta** determines the parts of the sentence, ie., subject, verb and object words which is inserted into the RDF records.

Document context is to be further limited by using semantic net software, sn**et**, which associates groups of words in a sentence with conceptual categories. I'm using use the software from the Open Mind common sense project called OMCSNetCPP to acquire the context for our docments.

So far the software **meta** and **snet** seem to be working okay. Our RDF storage software which maintains the document database is the Redland RDF store. Redland is a semantic graph database which uses SPARQL as its query tool.

These software tools are new and most of it is has not been tested well. But these tools have the potential to find intricate patterns in large, dynamically changing text data. The RDF triples representing the nebulous attributes of textual data makes determining text contents accurately a hard problem. Extracting meaning by finding patterns in portions of a large graph, that is by pattern matching subgraphs, makes the finding meaning easier.
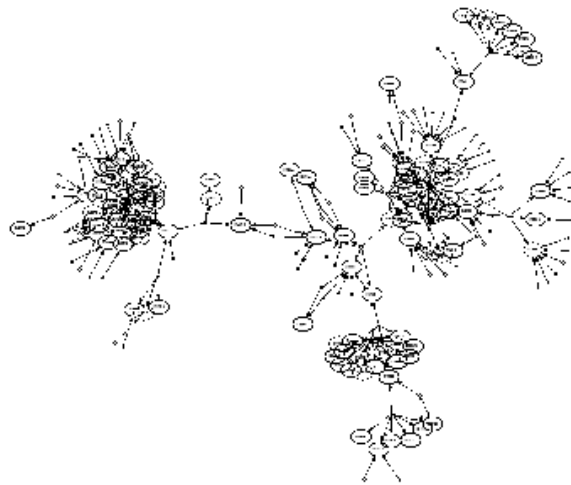

Figure 3: **Large RDF Graph**

---

another method of acquiring context. Using metadata as object-oriented data structures which can attribute elements of itself to the data and other metadata objects enables the mechanical data extractor to determine the context to a limited extent.

3 http://www.d.umn.edu/~tpederse/pos.html. "In (Brill 1992), a trainable rule based tagger is described that achieved performance comparable to that of stochastic taggers.." -- Eric Brill

Software Components

The Metadata Extractor, Meta

The metadata creator is the stepstone of our software tools. It creates the RDF records from reading the textual datasets extracting relationships and specific noun-entities as the metadata represent of text files. **Meta** uses Eric Brill's transformation base learning algorithm to identify the part of speech of each word in a sentence. The Brill POS is supplemented with Lance Ramshaw's noun phrase chunker so that phrases outside of the SPO relationship can be identified.

Meta is required to read text datasets efficiently with a set of 10,000 ontology profiles containing more than a total of 100,00 keywords. I'm hoping that the fully developed **meta** program will be able to read text data, and create RDF records, at a rate of 10 pages per second (on average about one document or message per second.) **Meta** runs as a server using soap web services.

We will need to tailor **meta'**s lookup tables for our applications.

SNet[4]

The **snet** software is Elliot Turner's version of OMCSNet[5] written in C++. I wrote a persistent client/server

---

4 A semantic network is composed of a collection of elements, a category, in which the objects of the category, C, can be related to each other by specific maps. These maps can be generated by functional operators called functors, F, which can transform a category C into a category D.

Concepts can be represented symbolically as a composition of elements or objects connected by functional operators. Elements can be further subdivided into categories, and attributes can be assigned to each element or object.

$$F: x \rightarrow y$$

Inferences in semantic networks are based on graph reasoning methods like spreading activation (Collins & Loftus, 1975), structure mapping (Gentner, 1983), and network traversal. Graph-based reasoning is associative and not as expressive, exact, or certain as logical inferences. It is much more straightforward to perform, and useful for reasoning practically over text.

5 Context chains are to me analogous to the synfire chains in neural networks. Context is implied by a word and associated with other words by predicates. The functional form F: x->y is written as a predicate, subject, object triple: p (s, o), with implications of a first order logic.

"The contextual neighborhood around a word (node in a graph) is found by performing **spreading activation** (another term used in neural nets) from that source node, radiating outwardly to include other concepts. The relatedness of any particular node is not just a function of the number of links away it is, but also considers how many paths there are from that node to the source node, and the directionality of the edge.

Another basic type of inference that can be done on a graph is building inference chains: Traversing the graph from one node to another node via some path of connectedness. This is not logical inference per se but a simplification of modus ponens transitive reasoning. The FindPathsBetween() feature in the ConceptNet Practical Reasoning API supports building inference chains." [2]

wrapper for **snet** which uses soap web services and mod_soap for the Apache web server.

We will need to tailor **snet**'s verb or predicate tables to our application.

RDF Store and Query

We're tested the Redland [3] RDF store and SPARQL [4], but haven't loaded large datasets.[6]

The software that does the heavy duty CPU processing are written in C/C++ using the template C++ Standard Library. **Meta,** the POS tagger and **SNet,** the context chain server, are written with C++ templates. These modules are written to run in a distributed processing environment. The Redland database/sparql query components are written in C.

Some web services components such as the interactive SPARQL query program are written in Perl. Mod_perl in Apache enables the Perl/Python scripts to run in memory persistently.

References:

1. RDF Semantics: W3C Recommendation 10 February 2004; http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

2. OMCSnetCPP: http://www.eturner.net/omcsnetcpp/wordnet/, and ConceptNet: http://web.media.mit.edu/~hugo/conceptnet/

3. http://librdf.org/

4. SPARQL Query Language for RDF, W3C Working Draft 12 October 2004, http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/

Spreading Activation is implemented by building a data structure called a **context graph**, which is a network of document and term nodes. All document nodes are connected to the terms that occur in that document; similarly, every term node is connected to all of the document nodes that term occurs in. We search the graph by starting at a query node and distributing a set amount of energy to its neighbor nodes. Then we recurse, diminishing the energy at each stage, until this spreading energy falls below a given threshold. Each node keeps track of accumulated energy, and this serves as our measure of relevance.

6 2 Mb of UniPort data was loaded. Redland used 24 Mb of the disk (Berkeley DB files). Redland's use of indices is very inefficient creating an index 12 times the raw corpus. (While it seemed that Infact-3 was CPU bound, Redland as with most RDF stores is not CPU bound, but requires lots of disk space.) Problems will be encountered while we're loading endless amounts of triples representing text data. For example, we estimate we'll need about 50 triples to represent a small text document about 3 pages long (a unit text message). A million text messages will require up to 50 million triples. Geoff Chappell, Intellidimension (RDF Gateway), loaded 11Gb of UniProt RDF data from the Swiss Bioinformatics Institute containing 260 million triples. This process took several days using 38.2 GB on disk.

Sample SparQL Query Form

Sample query form generate using perl scripts with Apache mod-perl.

Figure 4: **Sample Query Form**