Knowledge-based and vertical-driven information retrieval

Gavin Matthews
Vertical Search Works
gmatthews@verticalsearchworks.com

Max Petrenko
Vertical Search Works
mpetrenko@verticalsearchworks.com

Abstract – The paper introduces the architecture and functionality of the knowledge-based information retrieval technology developed at Vertical Search Works. A large-scale language-independent ontology is used during indexing, query analysis, and document retrieval as part of a web-scale vertical search engine. Three specific areas are examined: the knowledge resource, its visualization and editing toolbox, and ways of fine-tuning the ontology to enable intrinsic consistency checks will be discussed; using the knowledge resource for indexing and query analysis purposes, concentrating on how the ontology-driven semantic latching and disambiguation module tackles challenges arising from ambiguous input; tools for evaluating the performance of knowledge-based information retrieval available at Vertical Search Works.

Keywords: information retrieval, ontology engineering, disambiguation, vertical search engine

1 Introduction

Vertical Search Works (VSW) maintains and develops the Excalibur semantic web search engine (inherited from Convera) and the Editorial Related Advertising system (inherited from FirstLight ERA). Together these use semantic and linguistic processing of web pages to provide vertical search portals and context-sensitive advert placement. Behind the scenes, Excalibur relies on a large scale general-purpose ontology (loosely informed by WordNet), extended for the various verticals that are supported. The ontology includes both subsumption and lateral relations, supports inheritance and reasoning, and is used to latch and disambiguate concepts within unstructured natural language text.

Matching adverts to articles is handled essentially as a search problem, where the query is the article on which the adverts appear, and the search corpus is the set of available adverts including landing page, ad title and description, and any other available metadata. The task is slightly different because the key information from the article is not the content per se, but what it implies about the likely demographic of the reader. This means that the interpretation of articles and adverts is asymmetrical.

2 Structure and management of knowledge resources

The section will outline the structure of knowledge resources maintained by Vertical Search Works and discuss engineering, visualization and maintenance principles as well as maintenance and quality assurance tools developed at Vertical Search Works.

2.1 VSW knowledge resources

The word "synset" is used here to refer to concepts represented in the ontology that may be referenced in text (the name is inspired by WordNet's synonym sets). Structurally, the ontology features a language-independent layer of concepts, a language-specific layer of expressions, topic-specific facets, and domain-specific verticals. Having verticals in the ontology allows concepts to be clustered without restructuring the core ontological hierarchy. Vertical clustering thus adds to the flexibility and adaptability of the knowledge resources to different applications.

Like many general purpose ontologies including SUMO, DOLCE, Cyc, OntoSem (see [5], [1], [3], [4]), the VSW ontology has an upper ontology of entities, processes, attributes and relations, with further mid-level subclasses, augmented with lateral links (e.g. has theme, acts in, is the author of, is an accessory for). To maintain an acceptable level of abstraction during document retrieval, top-level ontological branches are inaccessible to the disambiguation and indexing modules. There are over half a million known concepts, including not only collections, but many individual known entities.

The ontology incorporates a lexicon of over two million expressions. Each expression is annotated with
a language, sensitivity to stemming and folding (case, diacritic, and punctuation), and other such restrictions. Most expressions are in English or are universal (language independent labels like proper names).

The expressions in the VSW lexicon are primarily direct natural language, but some are composed of other synsets to reflect semantic compositionality in language. An expression can incorporate by reference either all synonyms for a synset, or all synonyms for the synset and its subsumed descendants. For example, one of the expressions on the concept for “wood floor” is “$$cpr.005QO $cpr.000ZW$”. Here “$$cpr.005QO” denotes the concept of wood and all its children (e.g. oak, pine), and “$cpr.000ZW” denotes synonyms of “floor” (e.g. “plancher” in French, “piso” in Spanish). Compositional expression can mix synsets and words; another expression on “wood floor” is “$cpr.000ZW made of $$cpr.005QO”.

The underlying representation of Excalibur’s ontology is a set of frame-and-slot flat files. These are managed in a source-control system in parallel to source code, and are branched and merged in the usual ways. Ontologists do interact with these flat files, but also have a variety of other tools for viewing and editing the ontology. The ontology is also translated into other languages, such as RDF.

**Verticals** are topic areas (e.g. medicine, food, finance, photography, boxing, etc.) that may be used for the following purposes:

1) Concept grouping: each concept may be in zero or more verticals. This is deduced from ontology links and subsumption reasoning;
2) Document classification: documents may be in zero or more verticals, depending on the verticals of the concepts they contain; document membership is ranked;
3) Search interface specialization: the query system can be configured to prefer in-vertical interpretation of ambiguous expressions, and to prefer in-vertical result documents.

The mechanism of document search and retrieval described in (3) also incorporates facet- and drilldown-based search options.

**Facets** are document classifications, mostly applied by a pattern matching language based on concepts and word lists.

**Drilldowns** are groups of concepts, members of which might usefully be offered as narrower searches in some search interface.

**Word and phrase lists** are compiled that are useful for purposes like document classification, and detection of spam and adult content.

### 2.2 VSW knowledge resource check and quality assurance toolbox

**Editing the ontology**

Ontologists modify knowledge resources in a number of ways: flat file editing, generating knowledge resource change files, and by using an online knowledge resource and management tool called SAGE.

**Flat file editing**

The most flexible and powerful way is simply to open the flat file representation in a text editor or to apply ad hoc editing scripts to those files. This is always available as a fallback mechanism, but is not the easiest way to perform day-to-day tasks. In particular, it is especially vulnerable to clashing with changes made by others. Source control merging is available and is, of course, invaluable, but is not semantic aware and cannot handle duplicate symbols and the like. The flat file format is frame-and-slot and does not directly represent additions and deletions.

**Generating Knowledge Resource change files**

For batch edits, and automatic symbol generation, there is a simple text format designed to represent additions and deletions of frames and slots. Such change files are often created by hand, but there are also several tools and scripts that can generate them. Because these change files represent an edit to the ontology and are less sensitive to edit conflicts, they are easier to polish over time and circulate for review.

**SAGE**

A web-based editing interface is also available. It can offer the frame-and-slot representation of some part of the ontology, such as some existing concept with all slots and inverse slots. The user can then add, remove, and modify slots, and even create new nodes (again with automatic symbol generation). Any difference between the set of assertions initially presented, and the set of assertions saved is taken to be the edit that the user intends. Such additions and deletions accumulate in a session that can be reviewed, abandoned, committed to source control, or passed on to another ontologist for review.

Underlying SAGE is a database reflecting the latest state of the ontology as submitted to source control, and a second database representing all the editing sessions and their respective modifications.
The history of any assertion or synset can be tracked over time, and proposed future changes can be detected. Simple reasoning can be performed against the ontology, either from the perspective of some point in the source control history, or through the lens of a specific session. An API is available for creating new sessions and populating them with changes from an external source.

This means that SAGE provides a framework for integrating human editing and review with any source of knowledge, and will be able to support such services as concept creation wizards, *ad hoc* intrinsic tests, and other ways to suggest changes automatically.

**Browsing and visualizing the ontology**

SAGE provides a way to browse the ontology, frame-by-frame, but a more powerful facility is embedded in every Query Front End (QFE). This additionally provides implementations of the special-purpose reasoning incorporated in the indexing and query pipelines, such as that for triggering verticals and facets, and ancestor latching. A key feature provided allows the user to enter a short document in a text field and immediately view how the text is processed by the tokenizer and stemmer, and all potential latches permitted by dictionary matching. Output may be formatted in HTML for human viewing, or in XML for use by external visualization tools.

The QFE provides multiple buttons to visualize an ontology node in various ways. These visualizations are provided using Graphviz [10], and are driven by data provided by the QFE’s debugging XML API. All ontology diagrams shown in this paper were taken directly from this tool. These always reflect the state of the knowledge resources in use by some search or indexing cluster. The QFE provides a SAGE “Edit” button on every synset that allows immediate lightweight correction of any issue by any engineer.

Figure 1 below illustrates the representation of the polysemous word “Java” in the knowledge resources with its three senses mapped to respective ontological concepts. The graph displays lexical and ontological subsumption. Relatively few horizont al links are shown here because the three focal concepts are not strongly related. The chains of subsumption BT (“broad term”) and BI noI (“uninheritable broad term”) links lead to top-level collections “physical object” and “mental object”, which are known to be disjoint (via the DJ link). BT_noI links point to top-level concepts that are excluded from inheritance, and

![Graph showing the representation of the polysemous word “Java” in the knowledge resources with its three senses mapped to respective ontological concepts.](image)

Figure 1: mid- and top-level ontological branching
the HOMY ("homonymy") links capture the threesense polysemy of the word "Java".

Consistency checks in knowledge resources

As part of assuring quality and coverage in the knowledge resources, intrinsic consistency checks are routinely performed, either as basic validation or as reports (which are permitted to have extant violations). Below is a concise list of basic principles that inform internal consistency check in the ontology. Consistent implementation of the principles should prevent unwarranted cyclicity, reflexivity and redundancy in vertical and horizontal link chains and also ensure minimal depth of concept description, sufficient partition of concept classes, and enable domain and range link restrictions.

Knowledge resource quality principles:

- No broken links (links to undefined nodes) or duplicate nodes are allowed.
- Every concept must have at least one explicitly defined horizontal link.
- Relations are irreflexive and anti-symmetric unless defined otherwise. Irreflexive relations defined as transitive are acyclic.
- Asserted links are not redundant with the inferential closure, such as subsumption links between a node and its grandparent.
- Collections defined as disjoint have no instances or sub-collections in common.
- Every node has a description and a parent description available in every supported language so that it can be described to the user.
- No pair of sibling concepts is ontologically (that is, ignoring the lexicon) indistinguishable.
- Links comply with any domain and range type constraints defined for the relation.
- Every indexable concept occurs at least once in the corpus.
- Every indexable concept falls into at least one vertical.
- Special identifiers (e.g. homepage URLs for known entities) are well-formed.

3 Using knowledge resources in information retrieval

The section will discuss the ontology-based architecture of document indexing, including disambiguation, and query analysis developed at Vertical Search Works.

3.1 Using ontology during indexing

During indexing, documents pass through a number of phases of processing: HTML parsing and boilerplate detection, language detection, tokenization and stemming, expression matching, disambiguation, dynamic entity extraction, and document classification.

Below is a brief outline of the indexing pipeline:

**HTML parsing and boilerplate detection:**
The HTML DOM (Document Object Model) is extracted, and the nodes are classified into core content and boilerplate. The text of the document is extracted from the core content.

**Language detection:**
Using character trigrams, the language of the document is established.

**Tokenization and stemming:**
The document text is converted into a stream of tokens. Each token may have variants depending on stemming, case-folding, diacritic-folding, and punctuation-folding. Token keywords and bigrams are assigned ranks.

**Expression matching:**
The token stream is compared to synset expressions, subject to any expression-specific matching restrictions. Candidate latches are identified.

**Disambiguation**
The disambiguation module serves to “latch” surface expressions to underlying ontological concepts and calculate reliable confidence and rank scores for each concept found in the document. Lexically ambiguous input may result in multiple concept candidates for a single input string. The latching, confidence and rank estimation rely on a window-based spreading activation network (a well-established procedure – for more details see [2] and [6]). The network is biased towards any vertical the document is known to be in a priori, the set of candidate latches from the entire document, and the set of candidate
latches in immediate proximity. A set of primary and secondary latches are selected and ranked.

Figure 2 above illustrates the “activation network” generated by propagating initial and progressively aggregated weights via ontological links. Resulting activation weights exceeding a threshold determine the “latching” to the selected concept, which is further indexed and also reinforces surrounding “latch” candidates. The input sentence is “Apple and Sun distributed new versions of Java”, and the ambiguous proper nouns are successfully “latched” to appropriate concepts (highlighted) by gaining mutual support from proximate “latch” candidates. The activation link chains, including link types and directions, are available for viewing via collapsible widgets. For the tokens “new” and “of”, two senses with case-sensitive expressions NEW for the synset “Network of Executive Women” and OF for the synset “oxygen fluoride” were initially considered as latch candidates but discarded due to the lack of activation support.

A key feature of this disambiguation process is that it is applied to all candidate latches, not just those that are known to be ambiguous in the knowledge resource. The need to be able to process a large amount of potentially unattested input (product, brand, titles, band names, etc.) during run time dictates a number of assumptions about the nature of data and the functionality of the text processing architecture. First, we assume that our KR will always be incomplete, because any otherwise unambiguous lexical item or expression may also appear as the title of a movie or the name of a band. Second, it is assumed that incomplete coverage in the knowledge resources should be supplemented dynamically, i.e. the system should have the functionality required to detect and resolve this situational ambiguity when no adequate expression could be found in the knowledge resources.

As a result, during text processing the system is given considerable leeway in selecting potential latch candidates and may even choose to ignore an expression available in the knowledge resources if there are strong contextual considerations against it. As a limited form of relevance blurring, ancestor synsets (collections and super-collections) of the selected latches are also added to the document as indirect latches. For example, if a document mentions "poodle" then we would recognize that the document was (at least weakly) about dogs, mammals, animals, etc.
Dynamic entity extraction

Dynamic entities such as telephone numbers, email addresses, and recipe preparation times are extracted automatically. The system not only detects a particular telephone number but also segments it into meaningful “roles”, i.e. area codes, domain names, etc.

More sophisticated entity extraction rules, developed in an in-house language, allow for programmatic gathering of named entities and concept-concept relations based on ordered sequences of concept expressions and key lexical items. To illustrate, in sentences like “Richard Reid was a member of Al-Qaeda” and “Jerry Stackhouse played for the Pistons” the membership relation between the individuals and respective groups would be extracted automatically based on the same triple rule. Triple rules are a powerful means to extract information from unstructured natural language text: resulting triples can further be stored in an RDF repository for reasoning, search, lookup, classification and retrieval purposes.

Document classification

The Excalibur indexer also executes a large number of facet rules. These are defined in a similar way as entity extraction rules, but the goal here is to provide high-level categorizations about the document. An example would be a facet rule detecting news article, academic article, FAQ, clinical guidelines that describe tobacco use in a positive light. Facet rules are very powerful, and can match syntactic sequences, mere proximity of terms, or specific parts of a document.

3.2 Using ontology during query analysis

The indexing process described above extracts the concepts underlying the natural language text of document, but the typically user query is also expressed in natural language. In the simple case where the query refers to a single concept within the application vertical, then it can be interpreted directly using an ontology-based lexicon. For more complex cases, disambiguation is required.

VSW’s patented approach ([7], [8]) is to use the query as a keyword search against a vertically-restricted corpus. The results of that search are then used to translate the query into synsets, or into a hybrid of synsets and residual keywords. The system examines the locations in the result documents where the query terms appear. These are scored for length, proximity, and corpus statistics. A semantic interpretation of the user query is then selected, possibly with alternative interpretations for interactive clarification dialogue. This approach allows Excalibur to interpret multiple concepts in the user query in accordance with their relation to each other and, when appropriate, to fill in missing words. There are other layers in the query pipeline concerned with business logic and presentation, but these are beyond the scope of this paper.

4 Performance evaluation and validation

The evaluation cycle of the ontology parallels that of Excalibur’s software development. It comprises five key elements: validation, black box end-to-end testing, glass box tests and desk-check QA, and reports.

Validation

This is a basic check of syntax and some simple inherent properties, such as acyclicity in transitive relations (see above). It is performed repeatedly throughout the cycle. Anyone editing the ontology will perform the test locally before committing changes to source control. In the same way that it checks that the source code will build, CruiseControl ensures that the ontology in source control passes validation. All components that use the ontology perform the same checks on every load, making it part of the indexing cycle.

Black box end-to-end

A key property of the ontology is that it performs well at its intended task within a software architecture. In order to test the search quality of the end-to-end system, Excalibur uses the TREC [9] approach of taking typical user queries and a set of documents that have been manually evaluated against those queries. These tests are not binary pass/fail, but rather measure the extent to which the system selects good documents as relevant and rejects bad documents, and to which it ranks good documents above bad ones.

Initially the GOV corpus was used, but this was subsequently supplanted by an internally-developed test corpus focused on typical user queries against typical web documents in the verticals of interest. Bad, good, and very good results are distinguished, and the votes of multiple evaluators are combined. The TREC measure “prefs_simp”, ignoring unevaluated documents, was found to be most useful. Results can be broken down by vertical and examined at the level of queries and individual search results. The low level of data means that fine-grained results must be used
with caution. The absolute results are less important than relative changes seen as the code and ontology are modified day-to-day.

The test corpus used for this end-to-end testing is included in every index, from development to production. The tests are executed daily against all indexes and the results reviewed frequently.

**Glass box semantic tests**

As part of the general tests of the Excalibur platform, there are automated pass/fail tests that evaluate specific semantic applications of the ontology, in performing query expansion, synset latching, or document classification. These are performed as part of daily builds, and every deployment of a production index. These tests are developed either as regression tests for issues resolved, or as functional tests of enhancements.

**Desk-check QA**

Desk-check QA (manual verification of completed bug fixes) is routinely performed as part of knowledge resource development and maintenance work at VSW. Most ontology revisions relate to reported bugs or enhancement requests, and are therefore subject to the usual software defect tracking and change management processes. They are therefore reviewed and tested in the same way as the software. Automated glass-box tests may be developed at this point. Desk-check QA practice has proven useful in identifying broader classes of problems from specific instances. For example, identifying missing classes of child concepts after observing isolated (i.e. reported separately) instances of reifying siblings has been instrumental in ontological acquisition.

**Reports and reviews**

A final area of quality assurance in the KR is that when issues are detected, the team brainstorms ways to detect other instances of the same issue automatically. Typically such new intrinsic tests will expose many existing problems (and benign exceptions), so they cannot immediately be made part of the core intrinsic tests used for validation. Instead they are first presented as reports, and used to sweep the ontology clean over time.

**5 Conclusion**

This paper described the large-scale ontology used by VSW as part of a web-scale semantic search engine. Unstructured documents and user queries are both assigned meaning using the lexical knowledge embedded in the ontology. Tools and process for knowledge acquisition, ontology maintenance, validation, and visualization were presented. In particular, certain principles were outlined for ensuring the consistency of the ontology and its usefulness for text processing. The process of disambiguation was discussed, with particular reference to relevant in-house visualization tools.

**6 References**


