5

A Case Study in Natural Language Based Web Search

Giovanni Marchisio, Navdeep Dhillon, Jisheng Liang, Carsten Tusk, Krzysztof Koperski, Thien Nguyen, Dan White, and Lubos Pochman

5.1 Introduction

Is there a public for natural language based search? This study, based on our experience with a Web portal, attempts to address criticisms on the lack of scalability and usability of natural language approaches to search. Our solution is based on InFact®, a natural language search engine that combines the speed of keyword search with the power of natural language processing. InFact performs clause level indexing, and offers a full spectrum of functionality that ranges from Boolean keyword operators to linguistic pattern matching in real time, which include recognition of syntactic roles, such as subject/object and semantic categories, such as people and places. A user of our search can navigate and retrieve information based on an understanding of actions, roles and relationships. In developing InFact, we ported the functionality of a deep text analysis platform to a modern search engine architecture. Our distributed indexing and search services are designed to scale to large document collections and large numbers of users. We tested the operational viability of InFact as a search platform by powering a live search on the Web. Site statistics and user logs demonstrate that a statistically significant segment of the user population is relying on natural language search functionality. Going forward, we will focus on promoting this functionality to an even greater percentage of users through a series of creative interfaces.

Information retrieval on the Web today makes little use of Natural Language Processing (NLP) techniques [1, 3, 11, 15, 18]. The perceived value of improved understanding is greatly outweighed by the practical difficulty of storing complex linguistic annotations in a scalable indexing and search framework. In addition, any champion of natural language techniques must overcome significant hurdles in user interface design, as greater search power often comes at a price of more work in formulating a query and navigating the results. All of these obstacles are compounded by the expected resistance to any technological innovation that has the potential to change or erode established models for advertising and search optimization, which are based on pricing of individual keywords or noun phrases, rather than relationships or more complex linguistic constructs.

Nevertheless, with the increasing amount of high value content made available on the Web and increased user sophistication, we have reasons to believe that a segment

of the user population will eventually welcome tools that understand a lot more than present day keyword search does. Better understanding and increased search power depend on better parameterization of text content in a search engine index. The most universal storage employed today to capture text content is an inverted index. In a typical Web search engine, an inverted index may register presence or frequency or keywords, along with font size or style, and relative location in a Web page. Obviously this model is only a rough approximation to the complexity of human language and has the potential to be superseded by future generation of indexing standards.

InFact relies on a new approach to text parameterization that captures many linguistic attributes ignored by standard inverted indices. Examples are syntactic categories (parts of speech), syntactical roles (such as subject, objects, verbs, prepositional constraints, modifiers, etc.) and semantic categories (such as people, places, monetary amounts, etc.). Correspondingly, at query time, there are explicit or implicit search operators that can match, join or filter results based on this rich assortment of tags to satisfy very precise search requirements.

The goal of our experiment was to demonstrate that, once scalability barriers are overcome, a statistically significant percentage of Web users can be converted from keyword search to natural language based search. InFact has been the search behind the GlobalSecurity.org site (www.globalsecurity.org) for the past six months. According to the Alexa site (www.alexa.com), GlobalSecurity.org has a respectable overall traffic rank (no. 6,751 as of Feb 14, 2006). Users of the site can perform keyword searches, navigate results by action themes, or enter explicit semantic queries. An analysis of query logs demonstrate that all these non-standard information discovery processes based on NLP have become increasingly popular over the first six months of operation.

The remainder of this chapter is organized as follows. Section 5.2 presents an overview of our system, with special emphasis on the linguistic analyses and new search logic. Section 5.3 describes the architecture and deployment of a typical InFact system. Section 5.4 is a study of user patterns and site statistics.

5.2 InFact System Overview

InFact consists of an indexing and a search module. With reference to Figure 5.1, indexing pertains to the processing flow on the bottom of the diagram. InFact models text as a complex multivariate object using a unique combination of deep parsing, linguistic normalization and efficient storage. The storage schema addresses the fundamental difficulty of reducing information contained in parse trees into generalized data structures that can be queried dynamically. In addition, InFact handles the problem of linguistic variation by mapping complex linguistic structures into semantic and syntactic equivalents. This representation supports dynamic relationship and event search, information extraction and pattern matching from large document collections in real time.

5.2.1 Indexing

With reference to Figure 5.1, InFact's Indexing Service performs in order: 1) document processing, 2) clause processing, and 3) linguistic normalization.

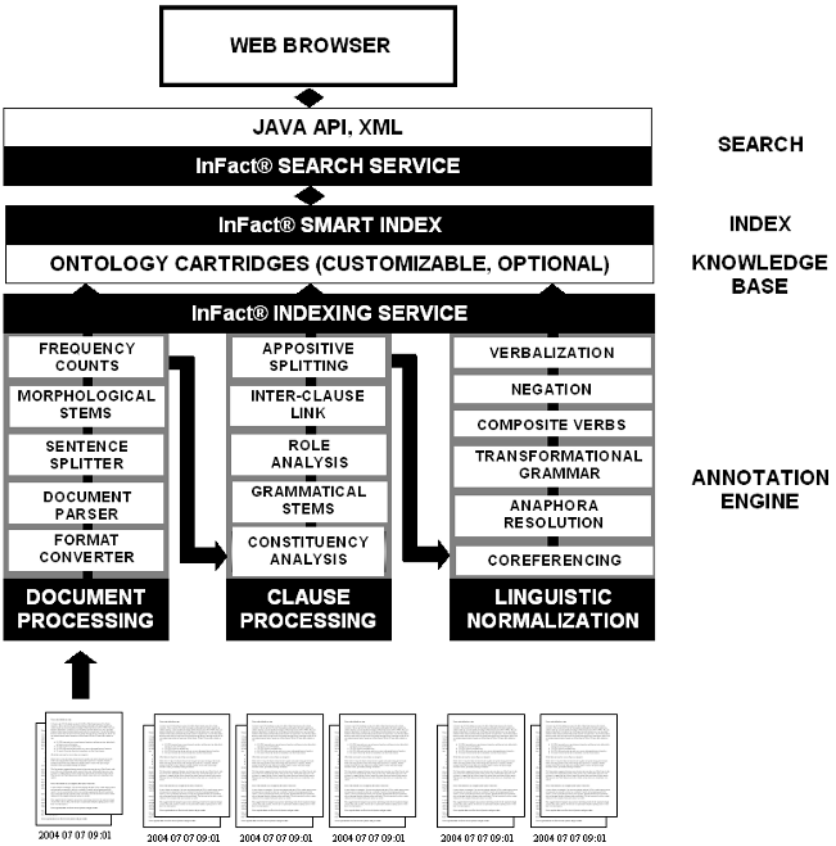


Fig. 5.1. Functional overview of InFact.

Document Processing

The first step in document processing is format conversion, which we handle through our native format converters, or optionally via search export conversion software from Stellant™ (www.stellent.com), which can convert 370 different input file types. Our customized document parsers can process disparate styles and recognized zones within each document. Customized document parsers address the issue that a Web page may not be the basic unit of content, but it may consist of separate sections with an associated set of relationships and metadata. For instance a blog post may contain blocks of text with different dates and topics. The challenge is to automatically recognize variations from a common style template, and segment information in the index to match zones in the source documents, so the relevant section can be displayed in response to a query. Next we apply logic for sentence splitting in preparation for clause processing. Challenges here include the ability to unambiguously recognize sentence delimiters, and recognize regions such as lists or tables that

are unsuitable for deep parsing. Last, we extract morphological stems and compute frequency counts, which are then entered in the index.

Clause Processing

The indexing service takes the output of the sentence splitter and feeds it to a deep linguistic parser. A sentence may consist of multiple clauses. Unlike traditional models that store only term frequency distributions, InFact performs clause level indexing and captures syntactic category and roles for each term, and grammatical constructs, relationships, and inter-clause links that enable it to understand events. One strong differentiator of our approach to information extraction [4, 5, 7, 8, 14, 19] is that we create these indices automatically, without using predefined extraction rules, and we capture all information, not just predefined patterns. Our parser performs a full constituency and dependency analysis, extracting part-of-speech (POS) tags and grammatical roles for all tokens in every clause. In the process, tokens undergo grammatical stemming and an optional, additional level of tagging. For instance, when performing grammatical stemming on verb forms, we normalize to the infinitive, but we may retain temporal tags (e.g., past, present, future), aspect tags (e.g., progressive, perfect), mood/modality tags (e.g., possibility, subjunctive, irrealis, negated, conditional, causal) for later use in search.

Next we capture inter-clause links, through: 1) explicit tagging of conjunctions or pronouns that provide the link between the syntactic structures for two adjacent clauses in the same sentence; and 2) pointing to the list of annotated keywords in the antecedent and following sentence. Note that the second mechanism ensures good recall in those instances where the parser fails to produce a full parse tree for long and convoluted sentences, or information about an event is spread across adjacent sentences. In addition, appositive clauses are recognized, split into separate clauses and cross-referenced to the parent clause.

For instance, the sentence: "Appointed commander of the Continental Army in 1775, George Washington molded a fighting force that eventually won independence from Great Britain" consists of three clauses, each containing a governing verb (appoint, mold, and win). InFact decomposes it into a primary clause ("George Washington molded a fighting force") and two secondary clauses, which are related to the primary clause by an appositive construct ("Appointed commander of the Continental Army in 1775") and a pronoun ("that eventually won independence from Great Britain"), respectively. Each term in each clause is assigned a syntactic category or POS tag (e.g., noun, adjective, etc.) and a grammatical role tag (e.g., subject, object, etc.). InFact then utilizes these linguistic tags to extract relationships that are normalized and stored in an index, as outlined in the next two sections.

Linguistic Normalization

We apply normalization rules at the syntactic, semantic, or even pragmatic level. Our approach to coreferencing and anaphora resolution make use of syntactic agreement and/or binding theory constraints, as well as modeling of referential distance, syntactic position, and head noun [6, 10, 12, 13, 16, 17]. Binding theory places syntactic restrictions on the possible coreference relationships between pronouns and

their antecedents [2]. For instance, when performing pronoun coreferencing, syntactic agreement based on person, gender and number limits our search for a noun phrase linked to a pronoun to a few candidates in the text. In addition, consistency restrictions limit our search to a precise text span (the previous sentence, the preceding text in the current sentence, or the previous and current sentence) depending upon whether the pronoun is personal, possessive, reflective, and what is its person. In the sentence "John works by himself," "himself" must refer to John, whereas in "John bought him a new car," "him" must refer to some other individual mentioned in a previous sentence. In the sentence, ""You have not been sending money," John said in a recent call to his wife from Germany," binding theory constraints limit pronoun resolution to first and second persons within a quotation (e.g., you), and the candidate antecedent to a noun outside the quotation, which fits the grammatical role of object of a verb or argument of a preposition (e.g., wife). Our coreferencing and anaphora resolution models also benefit from preferential weighting based on dependency attributes. The candidate antecedents that appear closer to a pronoun in the text are scored higher (weighting by referential distance). Subject is favored over object, except for accusative pronouns (weighting by syntactic position). A head noun is favored over its modifiers (weighting by head label). In addition, as part of the normalization process, we apply a transformational grammar to map multiple surface structures into an equivalent deep structure. A common example is the normalization of a dependency structure involving a passive verb form into the active, and recognition of the deep subject of such clause. At the more pragmatic level, we apply rules to normalize composite verb expressions, capture explicit and implicit negations, or to verbalize noun or adjectives in cases where they convey action sense in preference to the governing verb of a clause. For instance, the sentences "Bill did not visit Jane," which contains an explicit negation, and "Bill failed to visit Jane," where the negation is rendered by a composite verb expression, are mapped to the same structure.

5.2.2 Storage

The output of a deep parser is a complex augmented tree structure that usually does not lend itself to a tractable indexing schema for cross-document search. Therefore, we have developed a set of rules for converting an augmented tree representation into a scalable data storage structure.

In a dependency tree, every word in the sentence is a modifier of exactly one other word (called its head), except the head word of the sentence, which does not have a head. We use a list of tuples to specify a dependency tree with the following format:

(Label Modifier Root POS Head-label Role Antecedent [Attributes])

where: **Label** is a unique numeric ID; **Modifier** is a term in the sentence; **Root** is the root form (or category) of the modifier; **POS** is its lexical category; **Head-label** is the ID of the term that modifier modifies; **Role** specifies the type of dependency relationship between head and modifier, such as subject, complement, etc; **Antecedent** is the antecedent of the modifier; **Attributes** is the list of semantic attributes that may be associated with the modifier, e.g., person's name, location, time, number, date, etc.

For instance, the parse tree for our Washington example above is shown in Table 5.1.

Table 5.1. The parse tree representation of a sentence.

Label	Modifier	Root	POS	Head Label	Role	Antecedent	Attributes
1	Appointed	Appoint	V				
2	commander		N	1	Obj		Person/title
3	of		Prep	2	Mod		
4	the		Det	5	Det		
5	Continental Army		N	3	Pcomp		Organization/name
6	in		Prep	1	Mod		
7	1775		N	6	Pcomp		Numeric/date
8	George Washington		N	9	Subj		Person/name
9	molded	mold	V				
10	a		Det	12	Det		
11	fighting		A	12	Mod		
12	force		N	9	Obj		
13	that		N	15	Subj	12	
14	eventually		A	15	Mod		
15	won	win	V				
16	independence		N	15	Obj		
17	from		Prep	16	Mod		
18	Great Britain		N	17	Pcomp		Location/country

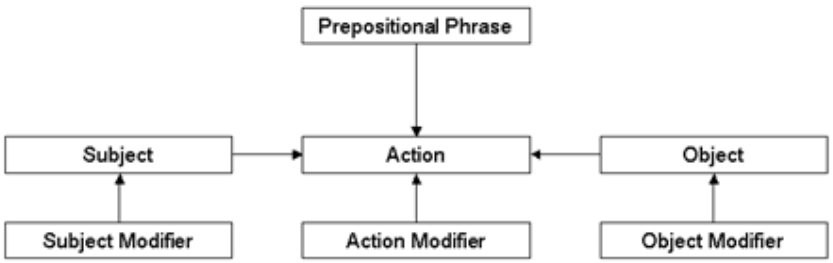


Fig. 5.2. The Subject-Action-Object indexing structure.

The basic idea behind or approach to indexing involves collapsing selected nodes in the parse tree to reduce the overall complexity of the dependency structures.

We model our storage structures after the general notion of subject-action-object triplets, as shown in Figure 5.2. Interlinked subject-action-object triples and their respective modifiers can express most types of syntactic relations between various entities within a sentence.

The index abstraction is presented in Table 5.2, where the additional column *"Dist"* denotes degrees of separations (or distance) between primary *Subject, Verb, Object* and each *Modifier*, and *"Neg"* keeps track of negated actions.

Table 5.2. The index abstraction of a sentence.

Subject	Subject-Modifier	Object	Object-Modifier	Verb	Verb-Modifier	Prep	Pcomp	Dist	Neg
		Washington	George	appoint				1	F
		commander		appoint				1	F
		Army	Continental	appoint				3	F
				appoint		in	1775	2	F
Washington	George	force	fighting	mold				2	F
force	fighting	independence		win				1	F
				win		from	Great Britain	3	F
				win	eventually			1	F

InFact stores the normalized triplets into dedicated index structures that

- are optimized for efficient keyword search
- are optimized for efficient cross-document retrieval of arbitrary classes of relationships or events (see examples in the next section)
- store document metadata and additional ancillary linguistic variables for filtering of search results by metadata constraints (e.g., author, date range), or by linguistic attributes (e.g., retrieve negated actions, search subject modifier field in addition to primary subject in a relationship search)
- (optionally) superimposes annotations and taxonomical dependencies from a custom ontology or knowledge base.

With regard to the last feature, for instance, we may superimpose a *[Country]* entity label on a noun phrase, which is the subject of the verb *"to attack."* The index supports multiple ontologies and entangled multiparent taxonomies.

InFact stores "soft events" instead of fitting textual information into a rigid relational schema that may result in information loss. "Soft events" are data structures that can be recombined to form events and relationships. "Soft events" are pre-indexed to facilitate thematic retrieval by action, subject, and object type. For instance, a sentence like "The president of France visited the capital of Tunisia" contains evidence of 1) a presidential visit to a country's capital and 2) diplomatic relationships between two countries. Our storage strategy maintains both interpretations. In other words, we allow more than one subject or object to be associated with the governing verb of a sentence. The tuples stored in the database are therefore "soft events," as they may encode alternative patterns and relationships found in each sentence. Typically, only one pattern is chosen at search time, in response to a specific user request (i.e., request #1: gather all instances of a president visiting a country; request #2: gather all instances of interactions between any two countries).

5.2.3 Search

Unlike keyword search engines, InFact employs a highly expressive query language (IQL or InFact Query Language) that combines the power of grammatical roles with the flexibility of Boolean operators, and allows users to search for actions, entities, relationships, and events. InFact represents the basic relationship between two entities with an expression of the kind:

Subject Entity > Action > Object Entity,

The arrows in the query refer to the directionality of the action, which could be either uni-directional (as above) or bi-directional. For example,

Entity 1 <> Action <> Entity 2

will retrieve all relationships involving *Entity 1* and *Entity 2*, regardless of their roles as subject or object of the action. Wildcards can be used for any grammatical role. For instance, the query "$*$ > *eat* > *cake*" will retrieve a list of anybody or anything that eats a cake; and a query like "*John* > $*$ > *Jane*" will retrieve a list of all uni-directional relationships between John and Jane. InFact also supports the notion of entity types. For instance, in addition to entering an explicit country name like "Argentina" as Entity 1 or Entity 2 in a relationship query, a user can enter a wildcard for any country name by using the syntax *[Country]*. InFact comes with a generic ontology that includes *[Location]*, *[Person]*, *[Organization]*, *[Numeric]* as the four main branches. Entity types can be organized hierarchically in a taxonomy. IQL renders hierarchical dependencies by means of taxonomy paths. For instance, in *[Entity/Location/Country]* and *[Entity/Location/City]* both *[Country]* and *[City]* nodes have a common parent *[Location]*. Taxonomy path can encode "is-a" relations (as in the above examples), or any other relations defined in a particular ontology (e.g., "part-of" relation). When querying, we can use a taxonomy node in a relationship search, e.g., *[Location]*, and the query will automatically include all subpaths in the taxonomic hierarchy, including *[City]*, *[Location]*, or narrow the search by expanding the path to *[Location/City]*.

With the InFact query language, we can search for:

- Any relationships involving an entity of interest

For example, the query "*George Bush <> * <> **" will retrieve any events involving "*George Bush*" as subject or object

- Relationships between two entities or entity types

For example, the query "*China <> * <> Afghan**" will retrieve all relationships between the two countries. Note in this case a wildcard is used in "*Afghan**" to handle different spelling variations of Afghanistan. The query "*Bin Laden <>*<> [Organization]*" will retrieve any relationships involving "Bin Laden" and an organization.

- Events involving one or more entities or types

For example, the query "*Pope > visit > [country]*" will return all instances of the Pope visiting a country. In another example, "*[Organization/name] > acquire > [Organization/name]*" will return all events involving a named company buying another named company.

- Events involving a certain action type

"Action types" are groups of semantically linked actions. For example, query "*[Person] > [Communication] > [Person]*" will retrieve all events involving communication between two people.

InFact's query syntax supports Boolean operators (i.e., AND, OR, NOT). For example, the query:

Clinton NOT Hillary > visit OR travel to > [Location]

is likely to retrieve the travels of *Bill Clinton*, but not *Hillary Clinton*.

We can further constrain actions with modifiers, which can be explicit entities or entity types, e.g., *Paris* or *[location]*. For example, the query

[Organization/Name] > buy > [Organization/Name]^[money]

will only return results where a document mentions a specific monetary amount along with a corporate acquisition. Similarly, the query

Bush <> meet<> Clinton ^[location]

will return results restricted to actions that occur in an explicit geographical location.

We can also filter search results by specifying document-level constraints, including:

- Document metadata tags – lists of returned actions, relationships or events are restricted to documents that contain the specified metadata values.
- Boolean keyword expressions – lists of returned actions, relationships or events are restricted to documents that contain the specified Boolean keyword expressions.

For instance, a query like:

[Organization/Name] > buy > [Organization/Name]^[money]; energy NOT oil

will return documents that mentions a corporate acquisition with a specific monetary amount, and also contain the keyword "energy" but do not contain the keyword "oil."

InFact also provides a context operator for inter-clause linking. Suppose for instance, that we want to retrieve all events where a plane crash kills a certain number of passengers. The event could be spread over adjacent sentences, as in: *"The plane crashed shortly after take-off. As many as 224 people were killed."*

In this case, a query like:

** > kill > [numeric] ~plane crash*

will retrieve all plane crash events, regardless of whether they are contained in a single or multiple, adjacent sentences.

InFact can also support synonyms and query expansion via custom ontologies. In this case, InFact will automatically recognize the equivalence of entities or actions that belong to the same ontology node.

The InFact Query Language rests on a flexible Java Search API. The Java Search API allows us to programmatically concatenate search operators, package and present them to the end user through a simpler interface.

5.3 Architecture and Deployment

We designed both indexing and search as parallel distributed services. Figure 5.3 shows a typical deployment scenario, with an indexing service on the left and a search service on the right. A typical node in each of the diagrams would is a dual processor (e.g., 2.8+GHz Xeon 1U) machine with 4GB of RAM and two 120GB drives.

The Indexing Service (left) processes documents in parallel. Index workers access source documents from external web servers. Multiple index workers can run on each node. Each index worker performs all the "Annotation Engine" analyses described in Figure 5.1. An index manager orchestrates the indexing process across many index workers. The results of all analyses are stored in temporary indices in the index workers. At configurable intervals, the index manager orchestrates the merging of all temporary indices into the partition index components.

A partition index hosts the actual disk based indices used for searching. The contents of a document corpus are broken up into one or more subsets that are each stored in a partition index. The system supports multiple partition indices: the exact number will depend on corpus size, number of queries per second and desired response time. Indices are queried in parallel and are heavily IO bound. Partition indices are attached to the leaf nodes of the Search Service on the right.

In addition to storing results in a temporary index, index workers can also store the raw results of parsing in a Database Management System (DBMS). The database is used almost exclusively to restore a partition index in the event of index corruption. Data storage requirements on the DBMS range between 0.5 and 6x corpus size depending on which recovery options for the InFact system are enabled. Once a document has been indexed and merged into a partition index it is available for searching.

In a typical search deployment, queries are sent from a client application; the client application may be a Web browser or a custom application built using the Search API. Requests arrive over HTTP and are passed through a Web Server to the Search Service layer and on to the top searcher of a searcher tree. Searchers are responsible for searching one or more partition index. Multiple searchers are supported and can be stacked in a hierarchical tree configuration to enable searching large data sets. The top level searcher routes ontology related requests to one or more ontology searchers, which can run on a single node. Search requests are passed to child searchers, which then pass the request down to one or more partition indices. The partition index performs the actual search against the index, and the result passes up the tree until it arrives back at the client for display to the user.

If a particular segment of data located in a partition index is very popular and becomes a search bottleneck, it may be cloned; the parent searcher will load balance across two or three partition indices. In addition, if ontology searches become a bottleneck, more ontology searchers may be added. If a searcher becomes a bottleneck, more searchers can be added. The search service and Web server tier may be replicated, as well, if a load balancer is used.

The example in Figure 5.3 is an example of a large-scale deployment. In the GlobalSecurity.org portal, we currently need only four nodes to support a user community of 100,000 against a corpus of several GB of international news articles, which are updated on a daily basis.

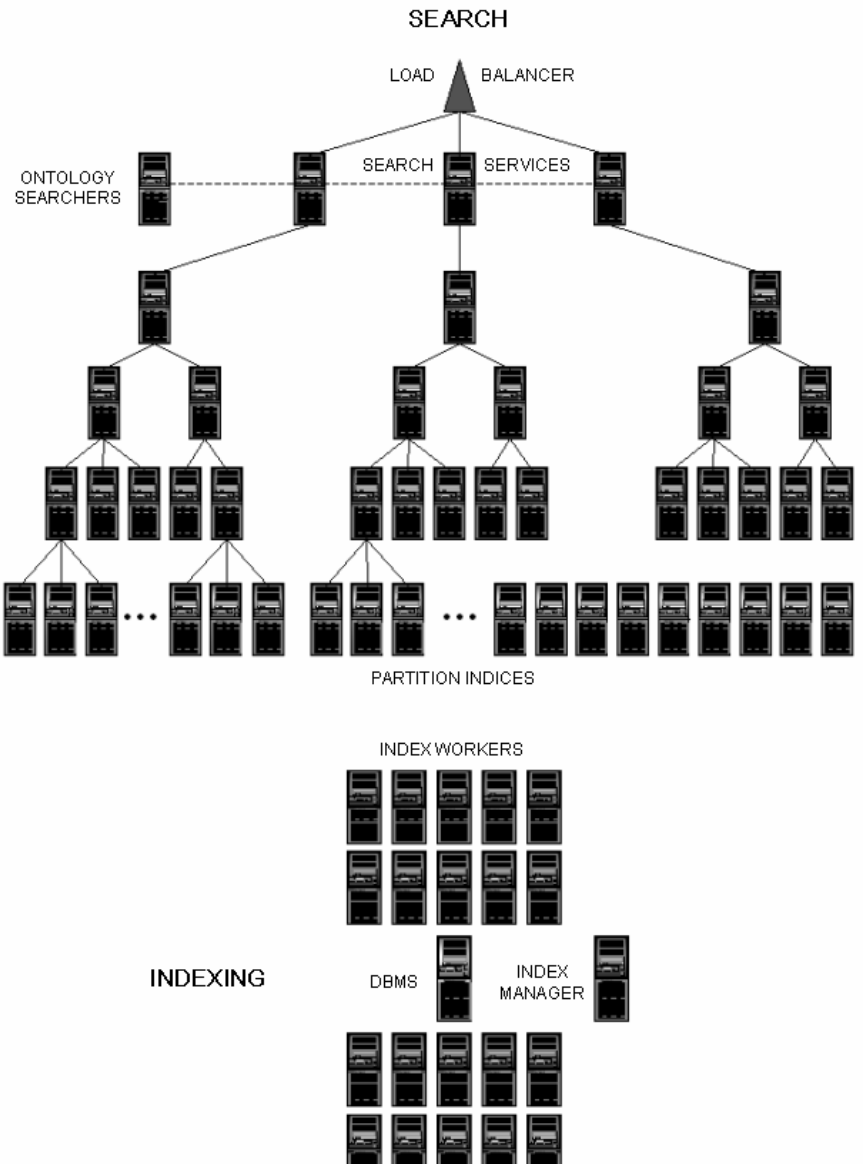


Fig. 5.3. Architectural overview of InFact.

5.4 The GlobalSecurity.org Experience

5.4.1 Site Background

InFact started powering the GlobalSecurity.org Web site on June 22, 2005. Based in Alexandria, VA, and "launched in 2000, GlobalSecurity.org is the most compre-

hensive and authoritative online destination for those in need of both reliable background information and breaking news ... GlobalSecurity.org's unique positioning enables it to reach both a targeted and large diversified audience. The content of the website is updated hourly, as events around the world develop, providing in-depth coverage of complicated issues. The breadth and depth of information on the site ensures a loyal repeat audience. This is supplemented by GlobalSecurity.org's unique visibility in the mass media, which drives additional growth" [9]. The director of GlobalSecurity.org, John Pike, regularly provides commentary and analysis on space and security issues to PBS, CNN, MSNBC, Fox, ABC, CBS, NBC, BBC, NPR, and numerous print and online publications. In powering this site, InFact serves the information search needs of a well-established user community of 100,000, consisting of news reporters, concerned citizens, subject matter experts, senior leaders, and junior staff and interns.

5.4.2 Operational Considerations

When preparing the GlobalSecurity.org deployment, one of our prime concerns was the response time of the system. For this reason, we kept the data size of the partition indices small enough so that most operations occur in memory and disk access is minimal. We split the GlobalSecurity.org data across two index chunks, each containing roughly 14 GB of data in each partition index. Another concern was having sufficient capacity to handle the user load. To account for future user traffic, we specified the deployment for 2-3 times the maximum expected load of about 11,000 queries per day. This left us with two cloned partition indices per index chunk. In addition, we wanted a hot back up of the entire site, in case of any hardware failures, and to support us each time we are rolling out new features.

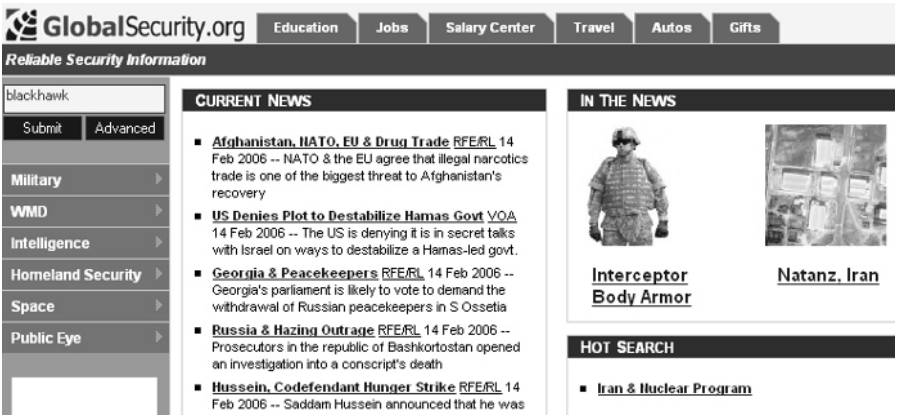


Fig. 5.4. The GlobalSecurity.org home page.

Another area of concern was the distribution of query types. Our system has significantly varying average response time and throughput (measured in queries/minute) depending on the type of queries being executed. We assumed that users

would take some time to migrate from keyword queries to fact queries. Therefore, we selected a very conservative ratio of 50/50 fact-to-keyword query types with a view to adding more hardware if needed. After automatically generating millions of query files, we heavily loaded the system with the queries to simulate heavy traffic using JMeter, a multi-threaded client web user simulation application from the Apache Jakarta organization. Based on these simulations, we deployed with only four nodes.

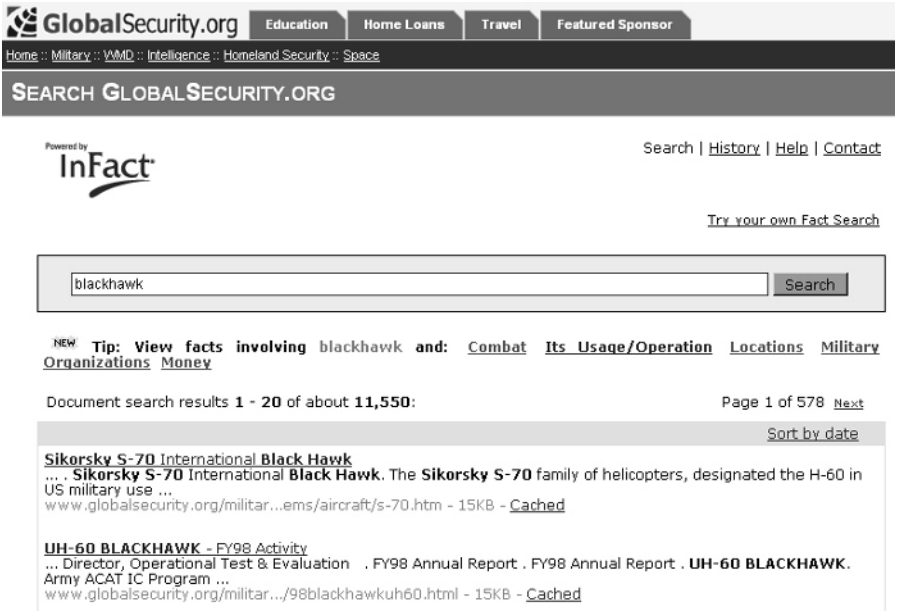


Fig. 5.5. Keyword search result and automatic tip generation with InFact in response to the keyword query "blackhawk."

5.4.3 Usability Considerations

In deploying InFact on the GlobalSecurity.org site, our goal was to serve the information needs of a wide community of users, the majority of which are accustomed to straightforward keyword search. Therefore, on this site, by default, InFact acts as a keyword search engine. However, we also started experimenting with ways to progressively migrate users away from keyword search and towards natural language search or "fact search." With reference to Figure 5.4, users approaching the site can enter InFact queries from the search box in the upper left, or click on the Hot Search link. The latter executes a predefined fact search, which is particularly popular over an extended time period (days or even weeks). The Hot Search is controlled by GlobalSecurity.org staff, and is outside the control of the general user. However, once in the InFact search page (Figure 5.5), the user can execute fact searches explicitly by using the IQL syntax. The IQL syntax is fully documented in the InFact Help page.

Alternatively, by clicking on the "Try your own Fact Search" link on the upper right of the InFact Search page, the user is introduced to a Custom Query Generator (Figure 5.6), which produces the query of Figure 5.7.

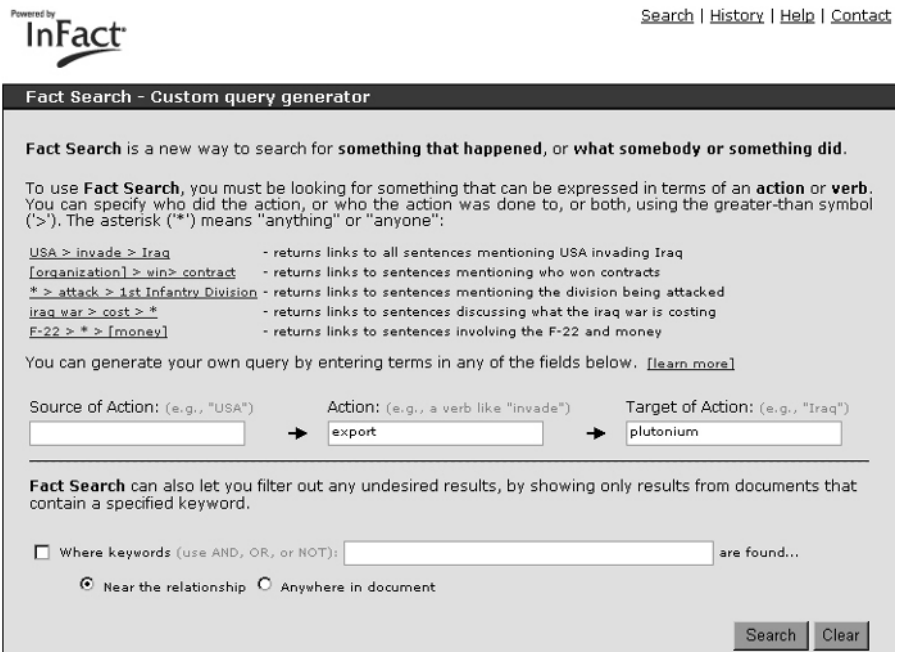


Fig. 5.6. Fact search with the InFact Custom Query Generator: the user is looking for facts that involve the export of plutonium.

The most interesting device we employed is guided fact navigation in response to a keyword entry. We call this process "tip generation." In this scenario, we capture keywords entered by a user and try to understand whether these are names of people, places, organization, military units, vehicles, etc. When executing a keyword search, the InFact system can recommend several fact searches which may be of interest to a user based on the keywords entered. These recommendations are presented to the user as a guided navigation menu consisting of links. In the example of Figure 5.5, the user is performing a keyword search for "blackhawk." The user sees a series of links presented at the top of the result set. They read: "Tip: View facts involving blackhawk and: Combat, Its Usage/Operation, Locations, Military Organizations, Money." Each of these links when clicked in turn executes a fact search. For instance, clicking on Military Organizations will generate the list of facts or relationships of Figure 5.8, which gives an overview of all military units that have used the blackhawk helicopter; clicking on Money will generate the list of facts or relationships of Figure 5.9, which gives an overview of procurement and maintenance costs, as well as government spending for this vehicle. The relationships are returned in a table display where each row is an event, and columns identify the three basic semantic

Fig. 5.7. Fact search with the InFact Custom Query Generator: InFact translates the query of Figure 5.6 into the InFact Query Language (IQL) and returns a list of results. IQL operators are fully documented in the Help page.

roles of source (or subject), action (or verb), and target (or object). Note that relationships, by default, are sorted by relevance to a query, but can also be resorted by date, action frequency, or alphabetically by source, action or target. Each of the relationships or facts in the table is in turn hyperlinked to the exact location in the source document where it was found, so the user can quickly validate the findings and explore its context (Figure 5.10).

Usage logs were the primary driver for this customization effort. The personnel at GlobalSecurity.org were very helpful and provided us with many months of user traffic Web logs. We wrote some simple scripts to analyze the logs. For example, we studied the 500 most popular key word searches performed on the site ranked in order of popularity. Next, we began looking for entity types that would be helpful to the most number of users. We found a lot of user interest in weapons, terrorists, and US officials, amongst other things. We then set about creating ontologies for each of these areas. New custom ontologies can easily be mapped into the internal InFact ontology XML format.

5.4.4 Analysis of Query Logs

We wish to quantify the relative popularity of natural language (Fact) search versus keyword search. In addition, we wish to compare the relative success of alternative strategies we adopted to overcome usability issues. This study of log data reflect

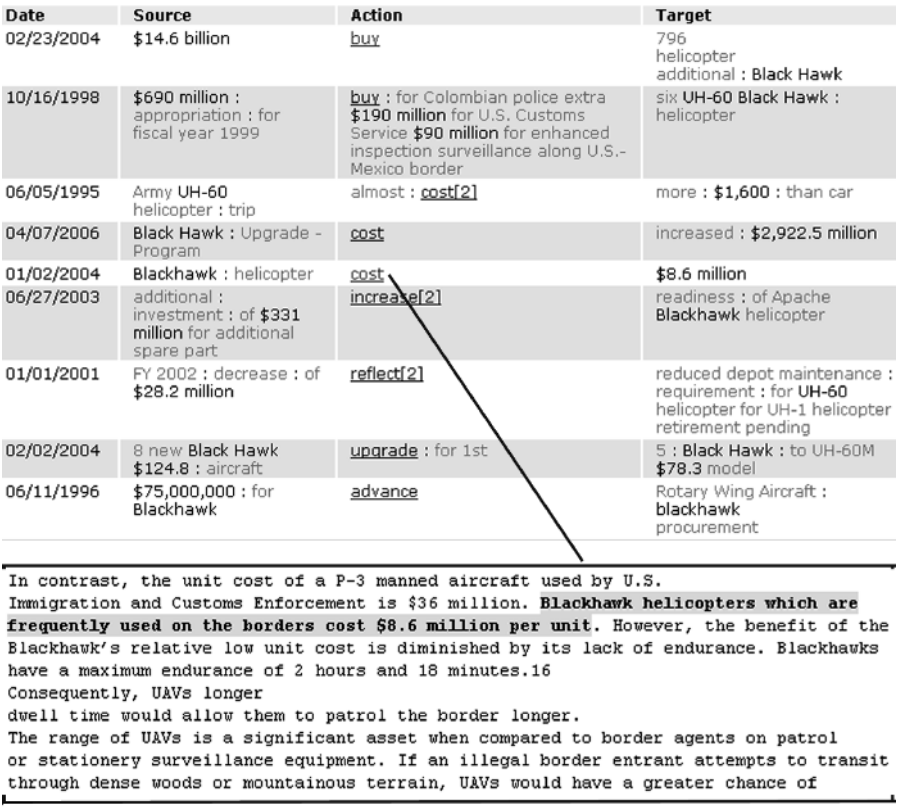
Chinook All : BLACK HAWK	include	160th Special Operations Aviation Regiment 101st Airborne Division 10th Mountain Division 82nd Airborne Division
Marine Corps	inspect : across airfield	rigging : on Blackhawk
U.S. Army	install	UH-60 : in future ALQ-211 : on CH-47
U.S. Army	introduce : for example	fly-by-wire : capability : to Army Apache Black Hawk fleet
air force	investigate	Black Hawk fratricide : incident
Troops : from 2nd Battalion 505th Parachute Infantry Regiment 82nd Airborne Division	kick off	Operation Desert Lion : with air assault from Chinook Black Hawk helicopter
Black Hawk : helicopter	land : As part of Operation Falcon Sweep	Shakaria Soldier : of 2nd Battalion 502nd Infantry Regiment
40 : CH-47 UH-60 AH-1	lift	1st Brigade : into
211th Aviation Group	maintain	UH-60 Blackhawk AH-64 Apache
air force : contractor	maintain	Blackhawk : helicopter Army : Apache

Fig. 5.8. Tip Navigation with InFact: facts involving the "blackhawk" helicopter and military organizations.

Date	Source	Action	Target
02/23/2004	$14.6 billion	buy	796 helicopter additional : Black Hawk
10/16/1998	$690 million : appropriation : for fiscal year 1999	buy : for Colombian police extra $190 million for U.S. Customs Service $90 million for enhanced inspection surveillance along U.S.-Mexico border	six UH-60 Black Hawk : helicopter
06/05/1995	Army UH-60 helicopter : trip	almost : cost[2]	more : $1,600 : than car
04/07/2006	Black Hawk : Upgrade - Program	cost	increased : $2,922.5 million
01/02/2004	Blackhawk : helicopter	cost	$8.6 million
06/27/2003	additional : investment : of $331 million for additional spare part	increase[2]	readiness : of Apache Blackhawk helicopter
01/01/2001	FY 2002 : decrease : of $28.2 million	reflect[2]	reduced depot maintenance : requirement : for UH-60 helicopter for UH-1 helicopter retirement pending
02/02/2004	8 new Black Hawk $124.8 : aircraft	upgrade : for 1st	5 : Black Hawk : to UH-60M $78.3 model
06/11/1996	$75,000,000 : for Blackhawk	advance	Rotary Wing Aircraft : blackhawk procurement

Fig. 5.9. Tip Navigation with InFact: facts involving the "blackhawk" helicopter and money.

four ways users submit a natural language query to InFact: 1) they click on the Hot Search link; 2) they click on a keyword tip; 3) they click on an example in the Query Generator or Help page; 4) they attempt to type an explicit relationship or fact search using the IQL syntax.



Date	Source	Action	Target
02/23/2004	$14.6 billion	buy	796 helicopter additional : Black Hawk
10/16/1998	$690 million : appropriation : for fiscal year 1999	buy : for Colombian police extra $190 million for U.S. Customs Service $90 million for enhanced inspection surveillance along U.S.-Mexico border	six UH-60 Black Hawk : helicopter
06/05/1995	Army UH-60 helicopter : trip	almost : cost[2]	more : $1,600 : than car
04/07/2006	Black Hawk : Upgrade - Program	cost	increased : $2,922.5 million
01/02/2004	Blackhawk : helicopter	cost	$8.6 million
06/27/2003	additional : investment : of $331 million for additional spare part	increase[2]	readiness : of Apache Blackhawk helicopter
01/01/2001	FY 2002 : decrease : of $28.2 million	reflect[2]	reduced depot maintenance : requirement : for UH-60 helicopter for UH-1 helicopter retirement pending
02/02/2004	8 new Black Hawk $124.8 : aircraft	upgrade : for 1st	5 : Black Hawk : to UH-60M $78.3 model
06/11/1996	$75,000,000 : for Blackhawk	advance	Rotary Wing Aircraft : blackhawk procurement

In contrast, the unit cost of a P-3 manned aircraft used by U.S. Immigration and Customs Enforcement is $36 million. **Blackhawk helicopters which are frequently used on the borders cost $8.6 million per unit**. However, the benefit of the Blackhawk's relative low unit cost is diminished by its lack of endurance. Blackhawks have a maximum endurance of 2 hours and 18 minutes.16 Consequently, UAVs longer dwell time would allow them to patrol the border longer. The range of UAVs is a significant asset when compared to border agents on patrol or stationery surveillance equipment. If an illegal border entrant attempts to transit through dense woods or mountainous terrain, UAVs would have a greater chance of

Fig. 5.10. Tip Navigation with InFact: each fact is hyperlinked to the exact location where it was found in the source document.

At the time of this writing, an average of 36% of advanced search users click on the hot search link "Iran and Nuclear program," which executes a predefined search like *"Iran* $> * \sim$ *nuclear."* However, it is difficult to assess what the user experience is like because in 80% of cases the user performs non-search-related tasks, and therefore we don't know how long they spent looking at the results. Note that users clicking on this link may not realize that they are going to a search engine page, since the link title is ambiguous. The results of this search are quite good, and still relevant. The hot search is an important entry point into our search site, as 36% of all the fact search queries executed came from this source. It seems likely that adding more of these hot search links or otherwise accentuating them on the page would significantly increase user exposure to natural language based queries.

Our analysis of query logs shows that keyword tips are the most effective way to migrate users to Fact Search. Users who click on tips frequently follow up with

queries of their own. Tip clickers also write better queries, probably because, after seeing the column display, they have a much better sense of how queries can be composed. Keyword tip clickers typically find the results engaging enough to spend an average of 1.5 minutes studying the results: 37% of users go back and click on more than one tip. Even better, 87% follow up by clicking on the "Try your own Fact Search" link and try their own query. All of the queries attempted are queries; 90% produce results; our follow up analysis suggests that for two thirds of these queries the results are relevant to the users search goals. In other words, users who click on the tips are extremely likely not only to try their own fact search, but also to pay enough attention to the format to write both valid and useful queries.

Examples in the Help File or Query Generator are largely ineffective at getting users to try Fact Search. Because the results returned by the examples usually do not necessarily relate to what the user wishes to search on, the column display is more of a distraction than an enticement to try Fact Search. However, those who go on to try Fact Search, after clicking on an example, have a better chance of writing good queries. Example link clickers are less likely to experiment with Fact Search or invest time learning how it works. Seventy-two percent of users end their session after clicking on one or more examples, not even returning to perform the keyword search that presumably brought them to the site in the first place. Of the 28% who did not leave the site after clicking an example, two thirds went on to try a Fact Search. Only 6% of users click on examples after having tried a Fact Search query on their own. Analysis of this user group suggests that examples have a place in the UI, but are not sufficiently compelling to motivate users to try Fact Search alone. However, this evidence does lend support to the hypothesis that users who see the column display are more likely to create valid queries: 60% of the users who click on examples and go on to write their own queries write valid queries and get results, which is still a much higher percentage than for users who blindly try to create queries.

About 75% of users who try Fact Search directly by using the IQL syntax, and without seeing the column display first fail to get results. Forty-five percent of users write invalid queries where nouns are inserted in the action field (the most common error). Another common error is specifying too much information or attaching prepositions to noun phrases. We can detect some of these errors automatically, and we plan to provide automatic guidance to users going forward. About 20% of query creators get impressive results. Most successful users get their queries right on the first shot, and, in general, seem unwilling to invest much time experimenting. Successful users are most likely expert analysts. In reproducing their searches and inspecting their results, we estimate that they have a positive impression of Fact search. In 75% of cases the results of Fact Search take direct the user quickly to the relevant parts of relevant documents, providing a deeper overview and faster navigation of content. However, in 25% of cases, expert users also write queries that return no results. Reasons for this include specifying too much information or including modifiers or prepositional terms in the verb field such as: "cyber attack," "led by," and "go to." In many cases users would be successful by just entering the verb. In some cases, users get lots of fact search results, but lack the experience to refine their query, so they simply go back to keyword search. We should try to communicate how queries can be modified further if there are too many results, perhaps by adding an ontology tag, or a context operator to the query syntax. For instance, the query *"Bush > meet > [person]"* could yield a large number of irrelevant results, if

a user is only interested in a list of diplomatic meetings. The query can be refined as *"Bush >meet > [person/name]."* In this case, the addition of an ontology tag restricts the number of meetings to those that are likely to involve named political personalities of some relevance. If the user is primarily interested in meeting that involve talks on nuclear arms control, the query can be further refined as *"Bush > meet > [person/name] ∼ nuclear arms control."* Similarly, the query *"[country] > produce > uranium"* can be turned into the query *"[country] > produce >[numeric] uranium"* if a user is after quantities of uranium that are being produced around the world. In general, we observe that users accustomed to keyword search believe that specifying more terms translates into more accurate results. In moving these users to Fact Search we must encourage them to start as simple as possible, since the IQL can express in two words what would take 20 lines using Boolean language.

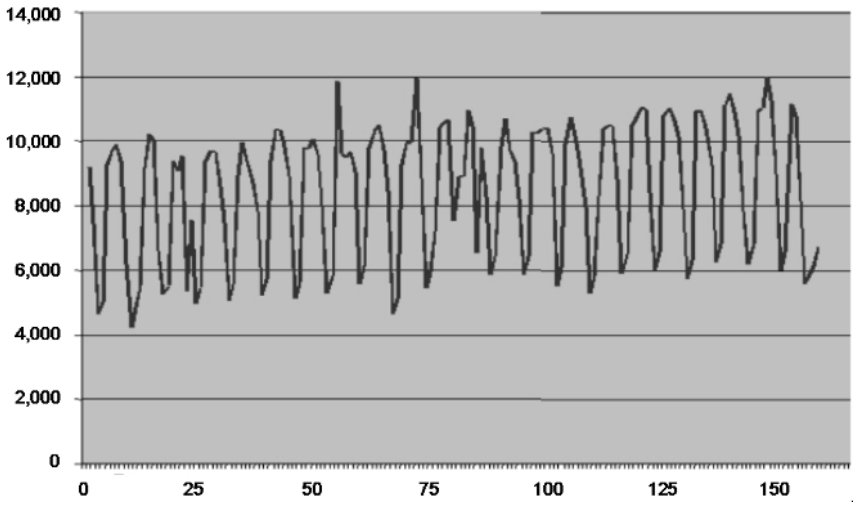


Fig. 5.11. Queries/day vs day of operation (June 22, 2005, to November 30, 2005).

Finally, Figure 5.11 shows overall query volumes (keyword search and Fact Search) as a function of day from the first day of operation (June 22 to November 30, 2005). The cyclic nature of the graph derives from the fact that most user access the site during the working week. Figure 5.12, which displays query volumes vs week of operation, clearly shows a positive trend: overall traffic to the site has increased by almost 40% ever since we introduced InFact search. The most interesting metrics relate to the percentage of users that derive value from Fact Search. The most effective mechanism to promote natural language search, as we have seen, are the tips. Figure 5.13 shows a 60% increase in the number of users that click on the tips automatically generated by InFact's advanced linguistic analysis over our entire period of operation. The overall percentage has increased from 4% to 10%. Our analysis also suggests that the best way to teach users how to write good queries is to first expose them to the summary result displays that ensues from a natural language query. The sooner users become aware of the type of results that a natural

language query can yield, the higher the chances that they learn how to use the new search functions correctly. This reinforces the idea that the result display may be a driver of Fact Search.

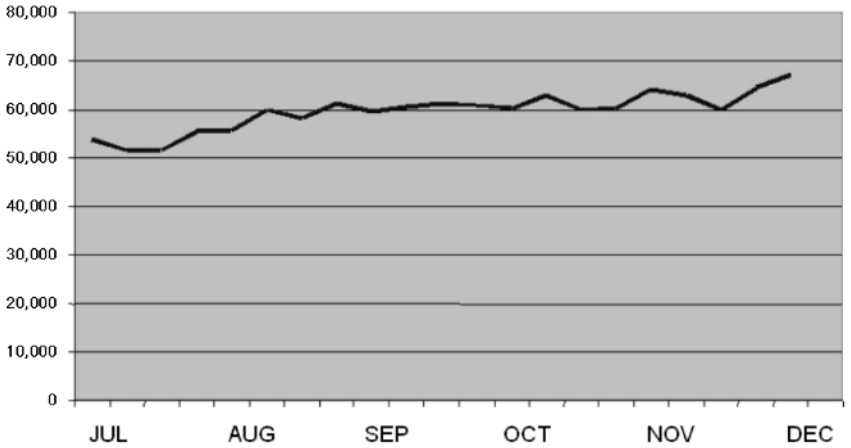


Fig. 5.12. Queries/week vs week of operation (June to November, 2005).

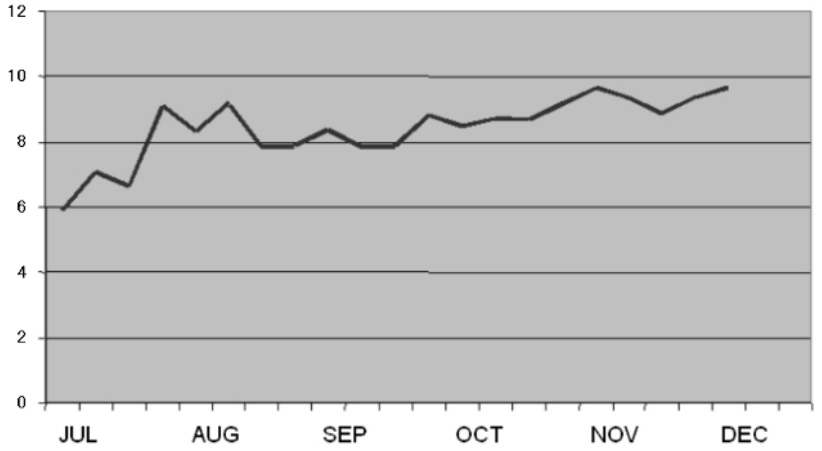


Fig. 5.13. Percentage of tips clicked (June to November, 2005).

5.5 Conclusion

We deployed a natural language based search to a community of Web users, and measured its popularity relative to conventional keyword search. Our work addressed criticisms of NLP approaches to search to the effect that they are not scalable and are too complex to be usable by average end-users. Our approach rests on a sophisticated index parameterization of text content, that captures syntactic and semantic roles, in addition to keyword counts, and enables interactive search and retrieval of events patterns based on a combination of keyword distributions and natural language attributes. Our distributed indexing and search services are designed to scale to large document collections and large numbers of users. We successfully deployed on a Web site that serves a community of 100,000 users. An analysis of query logs shows that, during the first six months of operation, traffic has increased by almost 40%. Even more significantly, we are encountering some success in promoting natural language searches. Our study demonstrates that the percentage of users that avail themselves of guided fact navigation based on natural language understanding has increased from 4% to 10% during the first six months of operation. Going forward, we will focus on increasing this percentage with a more innovative UI.

5.6 Acknowledgments

This work was partially supported by Dr. Joseph Psotka of the US Army Research Institute under contract No. W74V8H-05-C-0016. We are also indebted to John Pike, director of GlobalSecurity.org and his staff for providing us with many months of user traffic Web logs prior to going live.

References

1. D. Appelt and D. Israel. Introduction to information extraction technology. IJCAI-99 tutorial. http://www.ai.sri.com/∼appelt/ie-tutorial/ijcai99.pdf.
2. D. Appelt and D. Israel. Semantic approaches to binding theory. In *Proceedings of the Workshop on Semantic Approaches to Binding Theory. ESSLLI*, 2003.
3. A. Arampatzis, T. van der Weide, P. van Bommel, and C. Koster. Linguistically-motivated information retrieval. In M. Dekker, editor, *Encyclopedia of Library and Information Science*, Springer Verlag, volume 69, pages 201–222. 2000.
4. C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In C. Boitet and P. Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 86–90, San Francisco, California, 1998. Morgan Kaufmann Publishers.
5. I. Dagan, O. Glickman, and B. Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop Recognizing Textual Entailment*, 2005.
6. M. Dimitrov. A light-weight approach to coreference resolution for named entities in text. Master's thesis, University of Sofia, 2002.

7. D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.
8. D. Gildea and M. Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 239–246, 2002.
9. GlobalSecurity.org. http://www.globalsecurity.org/org/overview/history.htm.
10. M. Kameyama. Recognizing referential links: An information extraction perspective. In *Proceedings of the ACL'97/EACL'97 Workshop on Operation Factors in Practical, Robust Anaphora Resolution*, pages 46–53, 1997.
11. A. Kao and S. Poteet. Report on KDD conference 2004 panel discussion can natural language processing help text mining? *SIGKDD Explorations*, 6(2):132–133, 2004.
12. C. Kennedy and B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16^{th} International Conference on Computational Linguistics (COLING'96)*, pages 113–118, 1996.
13. S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.
14. D. Lin and P. Pantel. DIRT - discovery of inference rules from text. In *Knowledge Discovery and Data Mining*, pages 323–328, 2001.
15. C. Manning and H. Schutze. *Foundation of Statistical Natural Language Processing*. The MIT Press, 2000.
16. R. Miltov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'98)/ACL'98*, pages 869–875.
17. R. Miltov. Anaphora resolution: The state of the art. Working paper. University of Wolverhamption, 1999.
18. National Institute of Standards and Technology. Automatic content extraction (ACE). http://www.itl.nist.gov/iaui/894.01/tests/ace.
19. M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *41th Annual Meeting of the Association for Computational Linguistics*, pages 8–15, 2003.