SPS: A Web Content Search System Utilizing Semantic Processing

Joseph Leone
Dept. of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155 USA
Joseph.2.Leone@uconn.edu

Dong-Guk Shin
Dept. of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155 USA
shin@engr.uconn.edu

Abstract—**This paper describes a Web content search system that employs semantic processing. The system, called SPS (semantic processing system), consists of a crowd-sourced ontology, a component for updating and extending the ontology, an NL parser, a semantic matcher, and a content representation formalism called semantic processing (SP) logical form. A typical web search results in hundreds of pages. The user then carries out the tedious and daunting task of sifting through each page to find the relevant/interesting information. SPS aims to improve the relevance by building a layer of automated filtering on top of conventional search engines. SPS takes a user's natural language query, composes it into a keyword query, augments the keyword query with additional keywords, and presents it to the search engine. The query when augmented with additional keywords produces a richer search result set. SPS sifts through each search result page extracting grammatical and semantic information to compute page relevance. We present the architectural framework for SPS and also illustrate how it uses semantic processing to improve the quality of search results.**

Keywords-Web content mining; semantic processing; dynamic ontology development; collaboration system; information retrieval; biomedical literature mining

I. Introduction

One of the major problems with the Internet is its inability to find quality information with a higher precision. Web search engines produce either a list of too many items or a list of too few, with most of the items not relevant to users' interest/query. In many cases, the search outcome does include relevant items but currently, web surfers have the tedious and daunting task of sifting through web search result pages to find the relevant/interesting information.

The current manner of web searching can be divided into two phases: the "look" phase and the "find" phase. In the "look" phase a user presents keywords to the search engine and the search engine returns a set of pages the engine considers relevant to the user. In the "find" phase the user sifts through the search engine results to find the *actual* relevant/interesting information. We say *actual* because search engines either may not return any relevant information at all or the relevant information is buried somewhere in the collection of returned pages.

An examination of how look phase functions reveals why relevant information is usually buried. Look phase proceeds as follows. Using the user supplied keywords, a search engine retrieves pages that contain those keywords. The search engine then applies ad-hoc heuristics and machine learning techniques to the retrieved pages to compute their relevance. The heuristics that a search engine employs are word position, and utilization of HTML markup; Google uses the PageRank [1] algorithm. In Google's PageRank algorithm, linking determines relevance. The more links point to a particular page the higher Google believes the page to be relevant. If too few links point to a page, it will check whether the links are from pages that are deemed high quality (e.g., from universities, government offices, hospitals, etc.). The typical machine learning techniques search engines employ are word frequency, information gain, odds ratio, and Bayesian analysis on the text words.

Using heuristics and machine learning is helpful in computing page relevance. However, the relevance that search engines compute is generally not accurate because both heuristics and statistics-based machine learning techniques are unable to deal with

1. polysemy at the phrase level (e.g. "juvenile victims of crime" vs. "victims of juvenile crime")
2. synonymy at the word level, i.e. different words having almost the same meaning ("throttle" & "accelerator"; "road" & "street")
3. same words having different meanings ("soap bar" & "singles' bar")

In addition, heuristics and machine learning sometimes produce results for purely statistical reasons with no real "semantic" relevance to the user's query.

Given the innate ambiguity of expressing a query via keywords, the look phase provides less improvement potential. In contrast, the find phase has much greater potential for improvement/automation because it is presently expected to be carried out manually by a person. We argue that if Web search incorporates even partial natural language capabilities that could extract grammatical and semantic information from both the user's query and from visited pages, relevance could be computed more precisely and the quality of the search results would be greatly improved.

This paper proposes a semantic processing system (SPS) that improves Web search by increasing keyword quality during the look phase and automating the find phase. The intent of our approach is not to replace current search engines (e.g. Google), but to work in conjunction with them. The SPS is layered between the search engine and the human user.

Section II discusses related work. Section III details SPS, its components, and SPS logical form (or simply, *SP form)* which is the internal knowledge representation formalism used by SPS. Section IV shows an example of SPS processing a query against the biological literature. In the biological literature genomic structures, proteins, and other phenonena are generally described using natural language. We illustrate using examples why SPS outperforms traditional keyword search.

II. Related Work

The Web search community has been exploring use of semantic processing to improve the relevance of query results. Sieg et al. [16] proposed a semantic approach utilizing ontology-based user profiles to personalize the Web search. In their work, each user's search interests and preferences are modeled into ontological profiles. Unfortunately, this group's proposal to use the ontology is limited to organizing the user's context rather than modeling the general world knowledge. In utilizing the ontology to compute the relevance, this group uses conventional statistics methods. Another proposal for the use of semantic processing is SPARK by Zhou et al. [17]. Given a keyword query, SPARK outputs a ranked list of expanded queries which are obtained in three steps, term mapping, query graph construction and query ranking. The authors emphasize the novelty of query ranking, but this group's ontology construction is rather ad hoc and is far from the general knowledge representation frameworks originating from natural language process. Most recently, Shabanzadeh et al. [18] proposed expanding query using semantic relations. In this work, semantic relations are extracted from a lexical database called WordNet where semantic relations are basically limited to hypernymy/hyponymy (is-a relation) and meronymy/holonym (part-of relation). As such, this group does not exploit the wealth of techniques available from the natural language processing either.

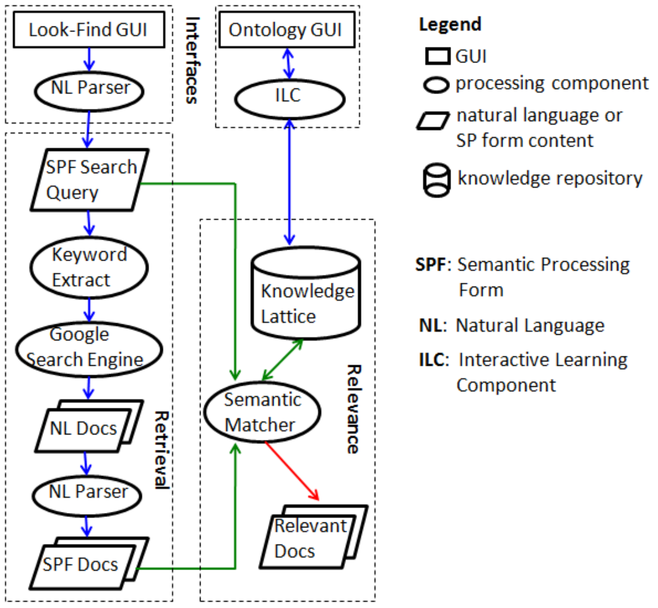

Figure 1. Semantic Processing System (SPS) architecture

III. System Architecture

In comparison with previous attempts, our proposed approach is more comprehensive by encompassing phrase parsing, knowledge-representation, query formation, ontology construction, and semantic matching. Figure 1 shows the three main parts (interface, retrieval subsystem, relevance computation subsystem) of the SPS architecture and their internal components:

SP form: A knowledge representation formalism used by SPS.

Canonical forms. Internal (not shown in Figure 1) lexical templates that specify every possible mapping from natural language to SP forms. Each canonical form represents a semantic pattern that occurs in English.

Parser for converting natural language to SP form.

Keyword Extract for creating a keyword query. The number of keywords in the search engine query may be greater than the number of words in the natural language query. The extra keywords account for synonymy and subtype/supertype.

Knowledge Lattice. A data structure for representing words, their subtype / supertype relationships, and their synonyms. Included in the data structure is a set of operations for reasoning about the relations between words.

Interactive Learning Component for updating and extending the Knowledge Lattice.

Semantic Matcher for computing retrieved pages' relevance.

SPS assists in two ways. During the look phase SPS accepts a natural language query, extracts from the query significant concepts, composes the concepts into a keyword search query that accounts for polysemy at the phrase level and synonymy at the word level, and presents the search query to a global search engine (e.g. Google). During the find phase, for each search engine result page, SPS semantically computes (using grammatical and semantic information extracted from retrieved pages and search query) the relevance of the result page to the user's natural language input query. This section is devoted to discussing five key computational aspects of SPS architecture.

A. Computable Representation

Every SPS architecture component, except NL Parser, is implemented in Lisp; and the core technology that underpins SPS, called *SP form*, is a computable internal knowledge representation expressed in Lisp notation. SP form is inspired by the semantics and logical foundation of Conceptual Graph (CG) [2,3]. However, an SP form looks very different from *conceptual graph interchange form* (CGIF) and only superficially resembles CG *linear form*. Moreover, many elements from CG's logical foundation (e.g., schematic cluster, prototypes, context, type definitions, aggregation, etc.) are not used. Nonetheless, some CG nomenclature (e.g., Knowledge Lattice, graph, subgraph, projection) is retained where there is great similarity at the abstract level between CG elements and SPS elements.

1) Canonical forms (i.e. semantic patterns) In their analysis of the English language Quirk et al. [4] determined that sentences are composed of clauses that in turn are made up of syntactic and semantic elements. Syntactic elements, i.e., subject, verb, object, complement, adverb, etc. are participants in the meaning of a clause. Semantic elements, i.e., agent, instrument, affected, etc. are the roles participants play [3,5]. Quirk identified thirty-three clause patterns, which account for all active English sentences.

For each of Quirk's clause patterns Leone [6,7] developed a *canonical form*. A canonical form is a conceptual graph lexical structure capable of representing the semantics of an active English sentence. Canonical forms are derived by mapping clause participants to concepts, and clause semantic roles to conceptual relations. A canonical form conveys information about the possible semantic roles of the participating syntactic constituents, provides predicates for representing each semantic feature, functions as a semantic pattern or template for capturing a particular class of meaning, and serves as a guide for mapping language (i.e. parser output) to logic (i.e. SP form). For each clause pattern, there is one and only one canonical form.

Below are some sample English language phrases, the phrase clause pattern, a canonical form corresponding to the clause pattern, and the phrase expressed in conceptual graph linear form. A complete set of clause patterns and canonical forms can be found in [6,7].

I bought her a gift.	S V O_i O_d
[c1] ← (agent) ← [c2] → (obj) → [c3] → (rec) → [c4]	
[I] ← (agent) ← [buy] → (obj) → [gift] → (rec) → [her]	

He put it on the shelf.	S V O_d A_{place}
[c1] ← (agent) ← [c2] → (obj) → [c3] → (loc) → [c4]	
[He] ← (agent) ← [put] → (obj) → [it] → (loc) → [shelf]	

He caught the ball.	S V O_d
[c1] ← (agent) ← [c2] → (obj) → [c3]	
[He] ← (agent) ← [catch] → (obj) → [ball]	

2) SP Form A sentence le*xical structure* consists of multiple phrases and each phrase is composed of a triple comprising a role and two participants. In SP form, each phrase is expressed as a role and two participants.

(<role> (<direction1> <participant1>) (<direction2> <participant2>))

The collection of such phrases (i.e., sp forms) constitutes a sentence.

Figure 2 shows the SP form syntax. The direction symbol → that points away from the role is read as "*is*", and the direction symbol ← that points to the role is read as "*of*". For example, *(manner (← eat) (→ fast))* is read as "manner of eat is fast".

<sentence>	:	(<phrase>+)
<phrase>	:	(<role> <participant> <participant> <participant>*)
<role>	:	function word (e.g., determiners, adverbs and prepositions) that clarifies relationships between concepts e.g., color, agent, location, obj, etc., i.e., conceptual relation
<participant>	:	(<direction><kernel>)
<direction>	:	← | →
<kernel>	:	<content> | (<content> <referent>)
<content>	:	content word (e.g., noun, adjective and verb) from catalog of conceptual types, e.g., dog, train, etc., i.e., concept
<referent>	:	<empty> | <individual> | <set> | <reference> | <measure> | <quantifier>
<individual>	:	a proper noun e.g., Snoppy, Clifford, Emma, etc.
<set>	:	(<individual>*)
<reference>	:	$
<measure>	:	@<number>
<number>	:	integer or floating point number
<quantifier>	:	@every

Figure 2. SP form Syntax

Participants could have a referent field. Figure 3 shows examples of participant referent field types, referent values, and their representation syntax in both CG and SP form. Figure 4 shows a natural language sentence expressed in SP form. Note that the sentence has three phrases.

Type	**CG**	**SP**
generic	[dog]	dog
generic set	[dog: {*}]	(dog (*))
individual	[dog: Snoopy]	(dog Snoopy)
set referent	[dog: {Snoopy, Lassie}]	(dog (Snoopy Lassie))
definite reference	[dog: #]	(dog $)
measure	[speed: @55]	(speed @55)
universal quantifier	[man: ∀]	(man ∀) (man @every)

Figure 3. Participant Referent Field Types

B. NL Parser: Stanford typed dependencies

The Stanford typed dependency (SD) [8] parser represents sentence grammatical relationships as typed dependency relations, i.e., triples of a relation between pairs of words, such as "the subject of *going* is *John*" in the sentence "John is going to Boston by bus". Each sentence word (except head of sentence) is the dependent of one other word. The dependencies are represented as *relation_name (<governor>, <dependent>)*. All are binary relations: grammatical relation holds between a governor and a dependent.

This representation, as triples of a relation between pairs of words, is well suited for mapping SD parser output to SP forms.

Figure 4 shows an SD parse of the sentence "John is going to Boston by bus". The parse output, which is the syntax tree and the SD dependencies, is mapped to SP forms via canonical forms. Note that for very complex sentences an additional parser that outputs parts of speech tags, for example [9,19], may be needed to determine the appropriate canonical form.

```
Parsing [sent. 1 len. 8]: [John, is, going, to, Boston, by, bus,.]
 (ROOT
   (S
     (NP (NNP John))
     (VP (VBZ is)
       (VP (VBG going)
         (PP (TO to)
           (NP (NNP Boston)))
         (PP (IN by)
           (NP (NN bus)))))
     (..)))
```

nsubj(going-3, John-1)
aux(going-3, is-2)
prep_to(going-3, Boston-5)
prep_by(going-3, bus-7)
	(agent (← go) (→ (person John)))
	(dest (← go) (→ (city Boston))
	(inst (←go) (→ bus))

Figure 4. Stanford dependency parser output and sp form

C. *Crowd-Sourced Knowledge Lattice & Interactive Learning Component*

Systems built using a knowledge-engineered approach suffer a disadvantage: they require much labor-intensive, difficult, manual work from a knowledge engineer in creating the ontology (i.e., explicitly defining every concept to be represented). The few general-purpose content languages that have been developed [10, 11] are usually bloated and unlikely to capture the intricacies of every possible domain.

This shortcoming is addressed by having SPS end users construct the knowledge lattice collaboratively. Will users provide extensive feedback? The Oxford English Dictionary, the world's greatest dictionary, was built in the 19th-century by a network of far-flung etymologists using postal mail. Today we have the Internet and the ideals of open source: share the goal, share the work, and share the results. Existing collaboration environments (e.g., Usenet News, eBay, Amazon reader surveys, epinions, tor, Foldit, Phylo, EteRNA) and community projects (e.g., Linux, GNU, Emacs, standards working groups, the Human Genome Project, raising a barn, etc.) demonstrate that users do provide feedback because of shared interest and perceived benefit.

The knowledge lattice resides on a central server and is accessible and modifiable by the Interactive Learning Component (ILC) of every SPS instance. The ILC is a teachable system [12,13] that acquires knowledge through dialog, and is responsible for maintaining and extending the knowledge lattice.

When SPS encounters a word not present in the Knowledge Lattice, SPS' Interactive Learning Component asks the user the position of the word in relation to other words in the lattice, and the word's synonyms if these are not available from a digital dictionary. Users are expected to indicate the position of the word in relation to other words. Updating the Knowledge Lattice may trigger other recursive Knowledge Lattice updates. The small incremental contributions from SPS' global population of users allow the Knowledge Lattice to be built quickly and with little effort from any one individual user. Figure 5 shows a Knowledge Lattice fragment and Figure 6 the lattice's computational representation.

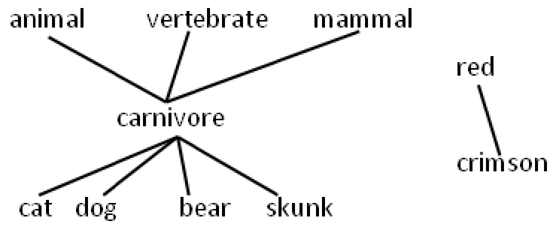

Figure 5. Knowledge Lattice Fragment

Word	Supertype	Subtype	Synonyms
carnivore	(animal vertebrate mammal)	(cat dog bear skunk)	(meat-eating flesh-eating predatory raptorial)
red	( )	(crimson)	(scarlet ruby cardinal flushed rosy wine sanguine)
crimson	(red)	( )	( )

Figure 6. Knowledge Lattice Internal Representation

The Knowledge Lattice stores no word definitions but only the subtype / super-type relations of a word and the word's synonyms. One key architectural decision of SPS is that computing relevance would not require use of "electronic" word definitions. In SPS, the relation of a word to other words in a phrase (e.g., role of agent, obj, etc.), the synonyms of a word (e.g., scarlet, red), and the subtype / super-type relation of a query word to a target word (e.g., person, girl) are used to determine if a target phrase matches (i.e., is relevant) a query phrase.

Now we introduce Knowledge Lattice operations. A lattice [2] is a structure consisting of a set L (of type labels), a partial ordering ≤, and two dyadic operators ∪ and ∩. If a and b are elements of L, $a \cap b$ is the *maximal common subtype* of a and b, and $a \cup b$ is the *minimal common supertype* of a and b. For any a, b, and c in L, these operators satisfy the following axioms:

- $a \cap b \le a$ and $a \cap b \le b$.
- If c is any element of L for which $c \le a$ and $c \le b$, then $c \le a \cap b$.
- $a \le a \cup b$ and $b \le a \cup b$.
- If c is any element of L for which $a \le c$ and $b \le c$, then $a \cup b \le c$.

Below are examples of Knowledge Lattice operations applied on the lattice fragment shown in Figure 5.

- ≤ subtype
  e.g., [crimson] ≤ [red]
- ∪ *minimal common supertype* i.e. least upper bound.
  e.g. [cat] ∪ [dog] = [carnivore]

[cat] and [dog] have many common supertypes including [animal], [vertebrate], and [mammal]. However, [carnivore] is the minimal common supertype.

- ∩ *maximal common subtype* i.e. greatest lower bound.

e.g. [vertebrate] ∩ [mammal] = [carnivore]

[vertebrate] and [mammal] have many common subtypes including [cat], [dog], [bear], etc. However, [carnivore] is the greatest common subtype.

D. Keyword Extract

Keyword Extract creates a keyword query for a traditional search engine (e.g., Google). Each keyword of the natural language query is augmented with additional keywords that account for synonymy and with its maximal common subtypes and minimal common supertypes. For example, if the natural language query is "*Carpets in Toyota cars cause accelerator to stick.*" and the Knowledge Lattice is as shown in Figure 7 then the generated keyword query contains all the synonyms, supertypes, and subtypes of each word in the natural language query. E.g., (*or* "accelerator throttle pedal choke car vehicle …").

Word	Supertype	Subtype	Synonym
accelerator	( )	( )	(throttle choke pedal)
car	(vehicle)	( )	(auto automobile motor-vehicle wheels clunker rustbucket)
carpet	( )	( )	(rug mat floor-covering blanket cover cloak)
cause	( )	( )	(root origin mainspring basis trigger foster make-happen create produce generate induce beget provoke)
stick	(wood)	( )	(cane pole club baton attach affix fasten paste pin tack stay last persist hold)
Toyota	(car automobile)	(Lexus Camry Corolla)	( )

Figure 7. Knowledge Lattice – car, carpet, cause

E. Semantic Matcher

Semantic Matcher (SM) determines which retrieved pages are relevant to the user's query. The inputs to SM are the user's query and the retrieved pages. SM carries out the relevance computation by applying the *restriction*, *projection*, *maximal-common-subgraph*, and *match* operations against the retrieved pages, each of which is explained below briefly. We note that these operations consult the Knowledge Lattice.

Restriction This operation transforms a concept into a more specific type. It replaces

- a more general concept with a more specific one, e.g. (animal) & (dog) => dog
- a generic referent with an individual referent e.g. (dog) & (dog Rufus) => (dog Rufus)
- individual/set-referent and set-referent with their union e.g. (dog Rufus) & (dog (Snoopy Lassie)) => (dog (Rufus Snoopy Lassie))
- two individuals with their union e.g. (dog Lassie) & (dog Rufus)) => (dog (Lassie Rufus))
- numerous other combinations for each participant type (see Figure 3), not listed due to space limitation

Maximal-Common-Subgraph This operation finds the largest subgraph that two graphs *u* and *v* have in common. For example,

u	*v*
(color (← (dog Rufus)) (→ brown)) (location (← (dog Rufus)) (→ porch))	(agent (→ dog) (← eat)) (obj (← eat) (→ bone)) (color (← dog) (→ brown))

The common subgraph is (color (← (dog Rufus)) (→ brown)) the generic referent (dog) is restricted to the individual referent (dog Rufus).

Match Two graphs *u* and *v* match if there is a subgraph *u*' of *u* such that

==> roles are the same in *v* and *u*'

==> pairs of corresponding concepts in *v* and *u*' have a *maximal common subtype*

For example,

u : "John likes white elephants."

v : "The boy likes animals."

u parser output:	*u* SP form:
nsubj(likes-2, John-1) amod(elephants-4, white-3) dobj(likes-2, elephants-4)	(agent (← like) (→ (person John))) (color (← elephant) (→ white)) (obj (← like) (→ elephant))

v parser output:	*v* SP form:
nsubj(likes-2, Boy-1) dobj(likes-2, animals-3)	(agent (← like) (→ boy)) (obj (← like) (→ animal))

Knowledge lattice operations indicate that (person) is a supertype of (boy), and (animal) is a supertype of (elephant); therefore, the graphs *u* and *v* match.

Projection This operation maps a general graph *v* to a more specialized graph *u*. The mapping is a subgraph of *u*, called a *projection* of *v* in *u*. A projection determines if a graph is a subgraph of another graph. E.g. adapted from [14].

u	*v*
(child (← (man John)) (→ (girl Mary))) (child (← (man John)) (→ (boy Bob))) (agent (← love) (→ (boy Bob))) (obj (← love) (→ (girl Mary)))	(child (← person) (→ person))

projection of *v* in *u*:
(child (← (man John)) (→ (girl Mary))) (child (← (man John)) (→ (boy Bob)))

Knowledge Lattice indicates that (person) is a super-type of (man), (boy), and (girl).

The difference between projection and match is that in match either graph can be specific or general; in projection, a more general graph is used to find a more specific one.

IV. Keyword Search vs Phrase Search

One specific application of SPS has been developing a phrase search mechanism for text mining of biomedical literature. Text mining of biomedical literature is a trendy topic in bioinformatics and we illustrate how SPS can improve relevancy in this application. Given below are two sentences from two different web pages [15]. These two pages are polysemous at the phrase level.

Page 1	We then present evidence that Sip1, Sip2, and Gal83 each interact independently with both Snf1 and Snf4 via distinct domains.
Page 2	The catalytic subunits of Arabidopsis SnRKs, AKIN10 and AKIN11, interact with Snf4 and suppress the snf1 and snf4 mutations in yeast.

A Google search against these pages with the query "interact with snf1" returns both pages 1 and 2 because all the search query keywords appear in both pages. But a SPS search returns only page 1 because the search query phrase occurs in page 1, but not page 2. This difference is the result of using the Semantic Matcher which utilizes the Knowledge Lattice and the Semantic Matcher operations to compute the relevance of page 2 to the user's query.

Figure 8 shows the SP form of each sentence. The search query in SP form is *(obj (← interact) (→ (protein Snf1)))*. This SP form search query matches the page 1 phrase *(obj (← interact) (→ (protein (Snf1 Snf4))))* but does not match any SP forms in page 2. There is a page 2 phrase, *(obj (← interact) (→(protein Snf4)))*, similar to the search query; but the page 2 phrase contains a different protein than the search query protein.

Page 1 sp form:	**Page 2 sp form:**
(agent (← present) (→ (person We)))	(type (← subunits) (→ catalytic))
(obj (← present) (→ $evidence))	(agent (← interact) (→ subunits))
$evidence:	(kind (← subunits) (→ SnRKs[1]))
(agent (← interact) (→ (protein (Sip1 Sip2 Gal83))))	(type (← SnRKs) (→ (plant Arabidopsis)))
(obj (← interact) (→ (protein (Snf1 Snf4))))	(equi (← SnRKs) (→ (protein (AKIN10 AKIN11))))
(manner (← interact) (→ independent))	
(instr (← interact) (→ domain))	(obj (← interact) (→ (protein Snf4)))
(type (← domain) (→ distinct))	
	(agent (← suppress) (→ subunits))
	(obj (← suppress) (→ mutation))
	(type (← mutation) (→ (protein (snf1 snf4))))
[1]SnRKs [Snf1 (sucrose non-fermenting-1)-related protein kinases	(loc (← suppress) (→ yeast))

Figure 8. Page 1 and 2 SP forms

V. Conclusion and Future Work

This example, albeit simple, demonstrates that phrase search produces results with higher relevance. Phrase search is superior because the atomic unit for matching is not a keyword but an inter-related collection of keywords, i.e. a phrase, that form meaning. The inter-relation expresses grammatical and semantic information, and is captured in SP forms. In contrast, in keyword search, the interrelation of keywords is only approximated by statistical quantities (e.g., word frequency, information gain, odds ratio, etc.) of the page containing the keywords.

We are currently examining the scalability of our method by applying the phrase search to flexibly finding gene regulatory relationships reported in the biomedical literature.

References

[1] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," www.db.stanford.edu/~backrub/google.html

[2] John F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, 1984.

[3] John F. Sowa and David Dietz, Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Pub Co, 1999.

[4] Randolph Quirk, et al. A Grammar of Contemporary English, Longman Group UK Limited, 1987.

[5] Fillmore, Charles J. (1968) "The case for case", in E. Bach & R. T. Harms, eds. Universals in Linguistic Theory, Holt, Rinehart and Winston, New York, pp. 1-88.

[6] J. Leone, "Synergistic Model for Memory Recall", Master's thesis, University of Connecticut, Storrs, CT, Aug. 1989.

[7] D. G. Shin and J. Leone, "AM/AG Model: A Hierarchical Social System Metaphor for Distributed Problem Solving," International Journal of Pattern Recognition and Artificial Intelligence, Vol. 4, No. 3, September 1990.

[8] http://www-nlp.stanford.edu/software/stanford-dependencies.shtml (last accessed 6/22/2011).

[9] http://www.connexor.eu/technology/machinese/demo/syntax/index.html (last accessed 6/22/2011).

[10] M. R. Genesereth and S. P. Ketchpel, "Software Agents", Communications of the ACM 37 (7), 1994, pp. 48-53.

[11] D. B. Lenat, "CYC: A large-scale investment in knowledge infrastructure", CACM 38(11), 1995, pp. 33–38.

[12] N. Haas and G. G. Hendrix, "An approach to acquiring and applying knowledge," Proc. of AAAI, 1980, pp. 235-239.

[13] N. Haas and G. G. Hendrix, "Learning by Being Told: Acquiring Knowledge for Information Management". In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), Machine Learning: An Artificial Intelligence Approach. Morgan Kaufmann Publishers, Inc., Vol. I, 1983, Chapter 13.

[14] Jean Fargues, et al., "Conceptual graphs for semantics and knowledge processing", IBM Journal of Research and Development, Vol. 30, No. 1, January 1986.

[15] http://www.ihop-net.org/UniPub/iHOP/gs/32484.html (last accessed 6/22/2011).

[16] A. Sieg, B. Mobasher, and R. Burke, "Learning Ontology-Based User Profiles: A Semantic Approach to Personalized Web Search", IEEE Intelligent Informatics Bulletin, Vol. 8, No.1. Nov. 2007.

[17] Q. Zhou, C. Wang, M. Xiong, H. Wang and Y. Yu, "SPARK: Adapting Keyword Query to Semantic Search", The Semantic Web (2007), Vol: 4825, Publisher: Springer, Pages: 694-707

[18] M. Shabanzadeh, MA Nematbakhsh, N. Nematbakhsh, "A Semantic based query expansion to search", in International Conference on Intelligent Control and Information Processing (ICICIP) 2010, IEEE, Pages: 523–528, (2010).

[19] http://www-nlp.stanford.edu/software/tagger.shtml (last accessed 6/22/2011).