Transactions on Machine Learning
and Data Mining
Vol. 6, No.2 (2013) 45-80
ISSN: 1865-6781,
ISBN: 978-3-942952-26-2

ibai Publishing
www.ibai-publishing.org

A Phrase-based Ontology Enabled Semantic Processing System for Web Search

Joseph Leone and Dong-Guk Shin

University of Connecticut, Department of Computer Science,
Storrs, CT 06269, USA

{Joseph.2.Leone, Shin}@uconn.edu

Abstract. Semantic processing system (SPS) is a system that performs phrase search of web content. SPS takes a user query in natural language, converts it to a keyword query, expands the keyword query with synonyms, hypernyms, hyponyms, and meronyms, and presents the keyword query to a search engine. SPS then sifts through the search engine result pages extracting grammatical and semantic information from each page for computing the page's relevance to the natural language query. SPS' relevance computation uses semantic matching of phrases rather than term-and-document frequency weighting—a method that is most commonly used by existing web search engines. SPS consults an ontology that is both "crowd-sourced," i.e., built collaboratively and incrementally by the large number of users and "auto-learned," i.e., contextually inferred from sentences containing desired words. SPS would be suitable for the areas of biomedical literature mining, legal document review and discovery, and news/RSS feed monitoring because these are laden with prose text. We implemented a prototype SPS, experimented with it and demonstrate that SPS outperforms a representative keyword based search engine. The strength of SPS stems from its exploitation of phrase semantics, which is not used in the conventional search engines.

Keywords: Web content mining; semantic processing; dynamic ontology development; collaboration system; biomedical literature mining.

1 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. This problem is further exacerbated in biomedical literature finding because search terms are scientific words that tend to be more phrase-oriented rather than a collection of multiple singular words.

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 the look phase reveals why relevant information is usually buried. Using 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 employed 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 few links point to a page, and those links are deemed high quality (e.g., from universities, government entities, hospitals, etc.), then the page is also considered relevant. The typical machine learning techniques employed by search engines are word frequency, information gain, odds ratio, and Bayesian analysis on the text words.

Using heuristics and machine learning methods are helpful in computing page relevance. However, the relevance measure 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"), and
3. *homonymy* at the phrase and word level, i.e. words having the same spelling but different meanings ("soap bar" & "singles' bar"; "north pole" & "ski pole")

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

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 [2]. We propose a semantic pro-

cessing 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 2 discusses related work. Section 3 describes SPS, its components, SPS logical form (or simply, *SP form*), the internal knowledge representation formalism used by SPS, and ontology construction in SPS. Section 4 shows SPS processing queries against the biological literature. Section 5 describes a method for evaluating SPS performance. Section 6 describes our experimental set-up and empirical results. Section 7 is the conclusion.

2 Related Work

Existing web search methods can be grouped into two categories: the *bag-of-words* hypothesis and the *semantic web* model. In the bag-of-words hypothesis [3] a document is viewed as a collection of words, and the frequencies of words in the document (i.e. bag) indicate the relevance of the document to a query (of keywords). Traditional search engines such Google, Bing, Yandex, Yahoo, etc. are based on the bag-of-words paradigm.

In the semantic web model, document words/content are not to carry any meaning. Instead another set of words that annotate the document are to signify the document's meaning. These other words are defined in an external ontology (RDFS[4]/OWL[5]), and "things" (e.g. people, places, events, music, movies, organizations, or just about any concept) mentioned in a document are annotated by Resource Description Framework (RDF) statements (also called *RDF triples*) that refer to the ontology words [6]. RDF is based on the ideas that resources have properties and that statements, which specify the properties and values, can describe resources. RDF statements are formed by three parts, called "subject" (resource), "predicate" (property), and "object" (value). Each subject, predicate, and object is a Uniform Resource Identifier (URI) that points to a Web resource (i.e., ontology word). Search engines such as Swoogle [7], Falcons [8], SWSE [9], Watson [10], and GoWeb [11] are built on the semantic web model.

We suggest that SPS introduces a new category of search method, called *parsed-prose*. In the parsed-prose paradigm a document is viewed as a collection of inter-related sentences, a coarse-level parsing is performed on the document prose text, and the parse result in conjunction with an ontology is used to compute the document relevance to the search query. "Coarse-level" means that the parse ignores grammatical elements (i.e. passive, active, infinitival, etc.) that indicate fine distinctions in meaning. A SPS spinoff system called VisuText [12], more clearly illustrates this parsed-prose paradigm.

We further summarize and compare these three paradigms in **Table 1**. One important note we emphasize is that SPS does not concern the Semantic Web. We elaborate some of the differences between SPS and the Semantic Web (SW).

Table 1. Paradigms of Information Retrieval

Paradigms/Models of Information Retrieval			
Paradigm	**parsed-prose**	**Semantic Web**	**bag-of-words**
Representative search engines	SPS	Swoogle [7], Falcons [8], SWSE [9], GoWeb [11], Watson [10]	Google, Yandex, Bing, Yahoo, Duck-DuckGo
Search query format	Natural language phrase or sentence	Keyword	Keyword
Operates over	Natural language prose text	Constructed language (RDF, RDFS/OWL, RDFa [13,14]) consciously devised for annotating WWW content with metadata.	Text words, html/xml markup, anchor text, microdata [15], microformats [16]
Linguistic meaning	Differential	Referential	Statistical
Primary carrier of meaning	Words in the context of natural language prose text	Annotation (i.e., tags) in the form of RDF, RDFa	Words, markup, microdata [15], microformat [16]
Processing atomic unit	SP form	RDF triple	word
Ontology (number of)	One ontology covering all domains. Word sense determined by context of word in sentence.	Different ontologies for different domains and word senses. Currently 4,400 different ontologies exist.	n/a
Ontology extension	Human or AutoLearn, which extracts relations as it reads prose text, and updates ontology with learned relations.	Uses SWRL [17] to extend OWL [5] with new relations.	n/a. However, performs keyword expansion.
Ranking & Relevance	# of phrases in query that match # of phrases in target sentence.	Links-based analysis applied to Linked Data [18] entities.	Links-based analysis applied to web pages (i.e., PageRank [1])
Data aggregation	n/a. Works only on prose text. Ability to understand query intent being developed.	RDF enables data-integration over data sourced from arbitrarily many sources (in theory).	Performed by human.

SP Form (SPF) vs RDF triples. SPS uses internal knowledge representation formalism, called SP form (SPF), which will be formally defined in Section 3.1. On the surface SPF and RDF triples may look similar since both are composed of three parts. However they are very different in what their respective constituents were derived from and are intended to describe, and function. SPF describes and is derived from the lexical structure of natural language (i.e., English), and thus is a computable representation of English prose text. The SPF vocabulary (i.e., participants) comes from words, processed by SPS, which naturally occur in English prose text. A single distinct SPF participant (i.e. the analogue of a subject or object in RDF) is able to encode thirteen different kinds of meaning (see Table 2).

Table 2. Participant Referent Field Types

Type	**SP Participant (encoding)**	**English reading**
generic	dog	a dog
generic set	(dog (*))	dogs, some dogs
generic counted set	(dog (*) @5)	five dogs
individual	(dog Snoopy)	Snoopy, the dog Snoopy
specified set	(dog (Snoopy Lassie))	Snoopy and Lassie
definite reference	(dog $)	the dog
definite set reference	(dog (*) $)	the dogs
measure	(speed @55)	speed 55
universal quantifier	(dog ∀) (dog @every)	every dog
universal negative	(dog ~) (dog @no)	no dog
universal plural	(dog (*) ∀) (dog (*) @every)	all dogs
universal plural negative	(dog (*) ~) (dog (*) @no)	no dogs
fuzzy plural quantifier	(dog (*) @many)	many dogs

The SPF role (i.e. the analogue of a predicate in RDF) comes from prepositions and verbs, is extracted automatically from natural language, and describes the interrelation of participants within a single subunit (i.e. phrase) of a sentence. SPF roles are not assigned a priori by a person, but instead all naturally occurring prepositions and verbs are mapped automatically to a finite set of roles. Roles and participants reflexively determine each other's word sense. The SPF functional scope is a single natural language sentence. RDF is a constructed prescriptive language that is not naturally occurring, but has been devised and imposed (i.e., prescribed) by W3C committees. RDF/OWL operates not within documents, as is the case with SPF, but across documents. Each RDF triple element (subject/predicate/object) links to other triples which themselves link to more triples. These strongly interlinked datasets are often referred to as "linked data" [18] or "web of data", and the links go across web pages, blogs, web sites, countries, etc. This interlinking constitutes the Semantic Web, and creates essentially a web universe parallel to the web of HTML/XML documents.

Linguistic meaning. In SPS, word meaning is determined by differences a word has with other words in an internal ontology; in Semantic Web, by an extra-linguistic

component (i.e., an annotation); and in the bag-of-words model, by statistics collected from a large sample of text in which the word occurs.

SPS function vs SW function. One key difference between Semantic Web and SPS is the granularity in encoding semantics and consequently the way encoded semantics is used in web searching. RDF/OWL cannot be easily used to represent the meaning (i.e., a computable representation) of a single arbitrary natural language (NL) sentence within a document. If RDF/OWL were employed to perform the SPS function (i.e., express a NL sentence in a computable form and process the computable form) every word in a NL sentence would have to be annotated with metadata to express the sentence in a form computable by a Semantic Web based process. Moreover, to process the RDF/OWL sentence representation, SWRL [17] would also be needed because OWL's set of relational properties does not cover the full range of expressive possibilities for object relationships that can exist in natural language. The number of object relationships in natural language is limited only by human imagination. In summary, annotating every word in every sentence that potentially exists in the web would be both impractical and unwise.

Search Query & Data Aggregation. For the search query, "list all heads of state of EU countries", adopted from [19], SPS will find a document containing the query phrase or a semantically equivalent variation of the query phrase. A Semantic Web based search engine instead, according to the vision of SW, will integrate information from multiple sources: a list of EU member states from europa.eu with a list of head of state by country from rulers.org to produce a list of heads of state of EU countries. SPS unlike SW does not draw from world/background knowledge beyond the context of the sentence it is processing and the ontology words that sentence terms point to.

SW systems with SPS-like architecture, purpose, and function. GoWeb [11] is a semantic web based system that is most similar in architecture and function to SPS. The purpose of GoWeb is to answer questions in the biomedical domain. The purpose of SPS is general information retrieval. GoWeb, like SPS, relies on a classical search engine (Yahoo). In GoWeb a user's input query is submitted to Yahoo, which returns a list of web document textual summaries, called snippets. GoWeb then uses entity recognition techniques to map concepts from ontological background knowledge (i.e. GeneOntology [20] and MeSH [21]) to the snippets. The snippets are annotated with background knowledge concepts. The annotation information is used to induce a tree representation, and to rank the top background knowledge concepts involved in the query. The top concepts provide the query answer.

GoWeb has some of the elements of SPS: an external classical search engine, a tree representation of knowledge, and ranking results with respect to the user's query. However, GoWeb differs with SPS in that in SPS the ontology is underpinned by a static permanent tree representation; and therefore, creating supplementary data structure like snippets annotation would not be needed, as the ontological category of snippet sentence elements is identifiable using the SPS ontology.

A concrete example from [11] illustrates this point and also illustrates how SPS equipped with a text summarization facility (not discussed in this article due to space limitation) can produce the same/better results than GoWeb. If a user query is "*Which diseases are associated with wnt signaling?*" and retrieved documents contain statements such as "*wnt signalling is linked to cancer*", then SPS can easily answer the user's query by first examining the ontology and discovering that "linked" and "asso-

ciated" are synonyms and "cancer" is a subtype of "disease"; and then identify via the projection operation that "*wnt signaling*" is a transition phrase between two distinct sentences (i.e. the two sentences can be canonically strung together). Sections 4.2, 3.5.1, 4.3, and 3.7.4 explicitly describe how such processes can be carried out. Some details of text summarization are provided in VisuText [12], an SPS spinoff system.

3 System Architecture

Fig. 1 shows the four main parts (interface, retrieval subsystem, auto-learn subsystem, relevance computation subsystem) of the SPS architecture and their internal components. Each 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) [22, 23]. 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, SPS uses some CG nomenclature (e.g., Knowledge Lattice, graph, subgraph, and projection) since there is great similarity at the abstract level between CG elements and SPS elements. In this section we briefly describe SPS components. A more detailed description can be found in [2].

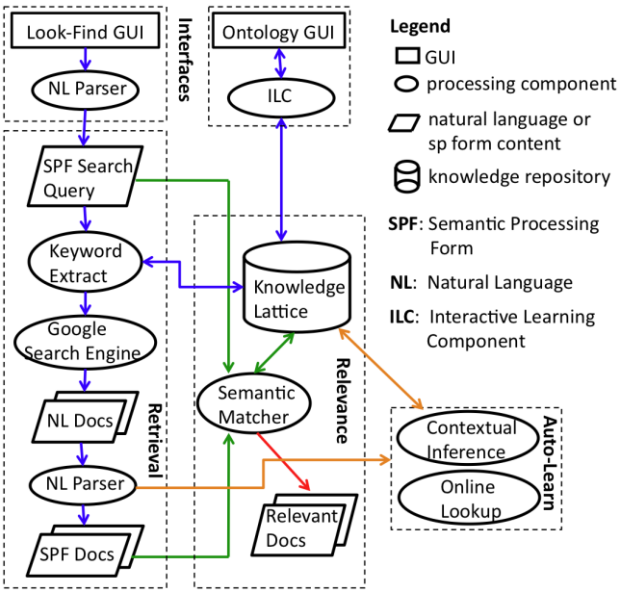

Fig. 1. Semantic Processing System (SPS) Architecture

3.1 SP Form

SP form [2] is the internal knowledge representation formalism used by SPS. An SP form expresses a sentence lexical structure in a computable format. A sentence consists of multiple phrases/clauses. Each phrase/clause is composed 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 *roles* participants play. Each phrase/clause is encoded in SP form as a triple comprising a role and two participants.

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

The collection of such phrases (i.e. SP forms) constitutes a sentence. Table 3 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, *(agent (← activate) (→ chemicals))* is read as "agent of activate is chemicals".

Table 3. SP Form Syntax

<sentence>	:	(<phrase>+)
<phrase>	:	(<role> <participant> <participant>)
<role>	:	function word (e.g., determiner, adverb, or preposition) 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, or verb) from catalog of conceptual types, e.g., dog, train, etc., i.e., concept
<referent>	:	ε | <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 | @many | @no

Participants could have a referent field. Table 2 shows examples of participant referent field types, referent values, and their representation syntax in SP form. Table 4 shows a natural language sentence expressed in SP form. Note that the sentence has three phrases.

For completeness Table 2 lists more referent types than are supported by SPS. SPS does not support *counted set*, *set reference*, *universal plural*, and *fuzzy plural*. These referent types express the exactness and granularity of meaning that is un-necessary for computing relevance. Also, for the sake of improving readability, in the examples that follow, the content is shown un-stemmed and the referent is omitted (e.g., "receptors" instead of "(receptor (*))").

3.2 NL Parser: Stanford typed dependencies

NL Parser implements the Stanford typed dependency (SD) [24] parser. The SD parser represents sentence grammatical relationships as typed dependency relations, i.e., triples of a relation between pairs of words, such as "the subject of *promote* is *receptors*" in the sentence "*Receptors promote chemicals in the cytoplasm*". 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.

Table 4. Parser dependency output and SP Form

Parsing [sent. 1 len. 7]: [Receptors, promote, chemicals, in, the, cytoplasm, .]	
nsubj(promote-2, Receptors-1)	(agent (← promote) (→ receptors))
dobj(promote-2, chemicals-3)	(obj (← promote) (→ chemicals))
det(cytoplasm-6, the-5)	
prep_in(chemicals-3, cytoplasm-6)	(loc (← chemicals) (→ cytoplasm))

The representation, as triples of a relation between pairs of words, is well suited for mapping SD parser output to SP forms. Table 4 shows an SD parse of the sentence "*Receptors promote chemicals in the cytoplasm*". The parse output, i.e. the syntax tree (not shown) and the SD dependencies (left column), is mapped to SP forms.

3.3 Crowd-Sourced Ontology

Systems built using a knowledge-engineered approach suffer from multiple disadvantages. They require much labor-intensive, burdensome, 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 [25, 26] are usually bloated and unlikely to capture the intricacies of every possible domain.

In SPS, this shortcoming is addressed by having end users construct the Knowledge Lattice (KL) collaboratively. We consider this model is very suitable for scientific literature mining like biomedicine because scientific knowledge is too vast for one person or a small group of scientists to curate all the known facts. A base Knowledge Lattice is initially built from words harvested from online resources such as dictionaries [27], encyclopedias [28] and UMLS [29]. Users then expand and update the base KL by adding new words or changing the relation of existing words. 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 can be a teachable system [30, 31] that acquires knowledge through dialog, and is responsible for maintaining and extending the knowledge lattice. Fig. 2 shows a Knowledge Lattice fragment and Table 5 the corresponding lattice's computational representation.

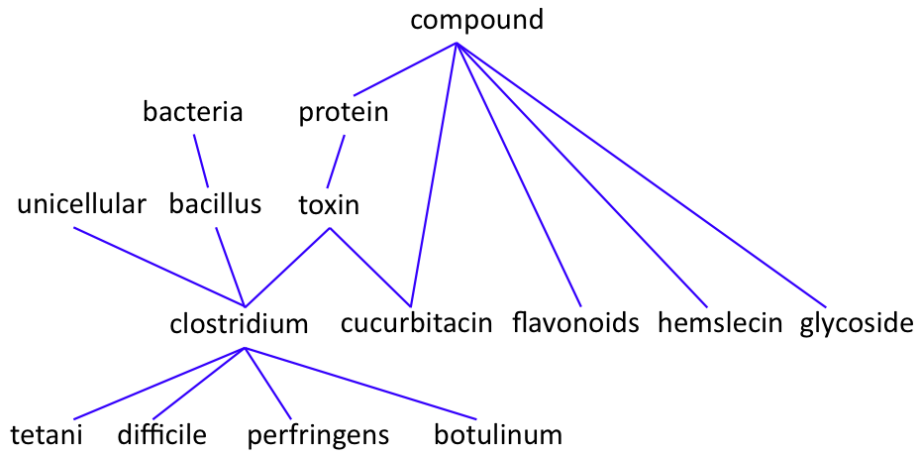

Fig. 2. Knowledge Lattice Fragment

Table 5. Knowledge Lattice Internal Representation

Word	**Supertype**	**Subtype**	**Synonyms**
bacteria	()	(bacillus)	()
unicellular	()	(clostridium)	(single-cell)
protein	(compound)	(toxin)	(enzyme)
toxin	(protein)	(clostridium cucurbitacin)	(hemotoxin phototoxin saxitoxin mycotoxin)
compound	( )	(protein cucurbitacin flavonoids hemslecin glycoside)	(mix amalgam composite complex blend combination)
clostridium	(unicellular bacillus toxin)	(tetani difficile perfringens botulinum)	( )

Interactive Learning Component - Learning by Being Told. 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 are to allow the Knowledge Lattice to be built quickly.

The ILC directs the user according to the following algorithm:

1. User is asked to add <*word*> to KL
2. User chooses from KL either a subtype or supertype of <*word*>
3. **if** subtype or supertype of <*word*> exists in KL
4. User places <*word*> in location appropriately
5. **else if** both subtype and supertype do not exist in KL
6. User suggests a subtype or supertype for <*word*>

The following example illustrates how a user would update the KL shown in Fig. 2. The user is asked to add "hemoglobin" to the KL. The user chooses a word in the KL that is a supertype or subtype of "hemoglobin". The user chooses "protein" which is a supertype of "hemoglobin". The user places "hemoglobin" below "protein" and draws an arc to "protein". If the user were asked to add the word "rickettsia" to the KL, the user would not find a supertype or subtype of rickettsia in the KL. In this case, the user might suggest "bacteria" as a supertype of "rickettsia".

Learn Area. Whenever a user extends the KL with a new word, the extension is used by SPS to process the user's query. However, the extension is not immediately incorporated into the KL or used for other queries; instead, the extension is stored in the Knowledge Lattice *learn area* (see Table 6). The learn area stores and counts the classifications of a particular word. The count (i.e., freq) is used to determine which classifications have the highest consensus. When a particular word has been classified 100 times, its highest frequency classifications are incorporated into the KL.

Table 6. Knowledge Lattice Learn Area

word	**freq**	**classification**
tetani	25	clostridium – tetani - ⊥
tetani	2	T - tetani – tick
tetani	3	T - tetani -⊥
tetani	65	bacteria – tetani - ⊥
tetani	5	protein – tetani - ⊥
protein	14	compound – protein - ⊥
bacteria	7	T - bacteria - ⊥
hemoglobin	1	protein – hemoglobin - ⊥
rickettsia	1	bacteria – rickettsia - ⊥

Table 6 shows user classifications of the words "tetani", "protein", "bacteria", "hemoglobin", and "rickettsia". Table 6 shows that "tetani" was classified 100 times, "protein" 14 times, "bacteria" 7 times, "hemoglobin" once, and "rickettsia" once. The figure also shows that "tetani" received five distinct classifications. Classifications are encoded as *"<supertype> - <word classified> - <subtype>"*. The symbols **T** and ⊥ are pre-defined KL elements: the universal type T represents *anything*, and the absurd type ⊥ is a proper subtype of every other type. All other elements are a proper subtype of the universal type.

ILC supports the operations *freq (<word>)*, which returns the number of times <*word*> has been classified and *distinct (<word>)*, which returns the number of dis-

tinct classifications of <*word*>. When *freq (*<*word*>*)* is at least 100, the top ranked classifications of <*word*> are incorporated into the KL. In Table 6, the classifications "bacteria – tetani - ⊥" and "clostridium – tetani - ⊥" of "tetani" are incorporated into the KL because they are the highest ranked "tetani" classifications.

3.4 Auto-Learn

In addition to interactive learning, SPS also employs *auto-learn*, a non-interactive learning mechanism. Auto-learn, which consisting of *contextual-inference* and *online-lookup*, is always attempted before interactive learning.

C*ontextual-inference* (CI) attempts to infer the meaning (i.e. position of word in the Knowledge Lattice in relation to other words) of an unknown word from the context of the word in a sentence. Contextual inference mimics the learning that occurs when children build vocabulary by reading. As children read they infer the meaning of unknown words from the context of the unknown word context present in the sentence, without having to look-up the word in a dictionary.

For example, when SPS encounters (i.e. reads) the following sentences: "*Woodward and his 17 year old sister Deanne, both of Niagara Falls, New York set out that day on a harmless boat ride on the upper Niagara River with family friend James Honeycutt. John Hayes, age 44 years, a truck driver from Voxhall, New Jersey was visiting Terrapin Point on Goat Island when he saw Deanne in the water.* " [32]

SPS' CI infers that "Niagara Falls" is both a city and waterfall in "New York", "Niagara" is a "River", "Honeycutt" and "Hayes" are a surnames, "Voxhall" is in "New Jersey", "Goat Island" is an "Island", and "Terrapin Point" is a place in "Goat Island".

CI discovers these facts by consulting both the KL and the dependency parse. From the dependency parse, CI is able to link a known fact with an un-known one, and update the KL with the new discovered facts. Note that CI determines word relatedness (i.e. linkage) from word dependency and not word proximity (as is often the case in machine learning approaches). CI also considers four types of modifier dependencies: adjectival, noun compound, appositional, and possession. The relations these word dependencies signify parallel the relational structure of the KL.

In the KL, edges represent variations in meaning between the two words that the edge connects. Edges capture three types of word relations: category-category (e.g. "vehicle" and "aircraft"), category-instance (e.g. "aircraft" and "Dreamliner"), and instance-instance (e.g. "Dublin" and "Ireland"). Category-category relations are generally found by interacting with the user or harvesting the relation from online dictionaries [27]. Category-instance and instance-instance relations are inferred from information encountered by SPS or by looking up the information in an online encyclopedia [28]. Note that instance-instance encompasses many types of relations such as "part of", "parts", "synonym", etc.

Contextual-inference provides three benefits: lessens the need for user interaction, enables "reading" online resources to extract new word meaning (i.e. relation of one word to another), and allows sentence in situ meaning extraction. These benefits contribute to the automatic construction of the KL.

Sentence in situ meaning extraction is particularly important in lessening user interaction because as SPS "reads" it may encounter new words that will not be in any dictionary, and their meaning (i.e. position in the Knowledge Lattice) may be known only to a person or can be contextually inferred from the sentence in which the word occurs. For example, in the sentence "*Rufus is the second dog to die at Petsmart grooming in six months*", "Petsmart", which is probably a dog beauty salon, will not be in any dictionary and its meaning will be known only to people who have knowledge of "Petsmart".

Online-lookup applies contextual-inference to online resources [28] to discover the meaning of unknown words.

3.5 Knowledge Lattice

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 words contained within a phrase (e.g. role of agent, obj, etc.), the synonyms of a word (e.g. protein, enzyme), and the subtype / super-type relation of a query word to a target word (e.g. toxin, recin) are sufficient to determine if a target phrase matches (i.e. is relevant) a query phrase.

Knowledge Lattice Operations. A lattice [22] is a structure consisting of a set L (of type labels), a partial ordering $\leq$, and two dyadic operators $\cup$ and $\cap$. 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:

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

Below are examples of Knowledge Lattice operations applied on the lattice fragment shown in Fig. 2.

- $\leq$ subtype
  e.g. [bacillus] $\leq$ [bacteria]
- $\cup$ minimal common supertype i.e. least upper bound.
  e.g. [difficile] $\cup$ [tetani] = [clostridium]
  [difficile] and [tetani] have many common supertypes including [unicellular], [bacillus], and [toxin]. However, [clostridium] is the minimal common supertype.
- $\cap$ maximal common subtype i.e. greatest lower bound.
  e.g. [bacillus] $\cap$ [toxin] = [clostridium]
  [bacillus] and [toxin] have many common subtypes including [difficile], [tetani], [perfringens], etc. However, [clostridium] is the greatest common subtype.

3.6 Keyword Extract & Expand

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. The data source of these additional keywords is the Knowledge Lattice. For example, if the natural language query is "*Compounds interfering with actin function*." and the Knowledge Lattice is as shown in Table 7, then the generated keyword query contains all the synonyms, supertypes, and subtypes of each word in the natural language query. E.g., (*or* "compound flavonoids cucurbitacin hemslecin glycoside blend mixture amalgam interfere hinder interact interlope meddle disrupt obstruct impede block actin protein filament function duty role purpose activity operate party").

Table 7. Knowledge Lattice - compound, interfere, actin, function

Word	**Supertype**	**Subtype**	**Synonym**
compound	( )	(flavonoids cucurbitacin hemslecin glycoside)	(blend mixture amalgam)
interfere	(hinder interact)	(interlope meddle)	(disrupt obstruct impede block)
actin	(protein)	(filament)	
function			(duty role purpose activity operate party)

3.7 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. These operations consult the Knowledge Lattice.

Restriction. *Restriction* transforms a concept into a more specific type. It replaces

- a more general concept with a more specific one, e.g. (bacillus) & (tetani) => tetani
- a generic referent with an individual referent e.g. (protein) & (protein myosin) => (protein myosin)
- individual/set-referent and set-referent with their union e.g. (protein myosin) & (protein (formin profilin)) => (protein (myosin formin profilin))
- two individuals with their union e.g. (protein profilin) & (protein myosin)) => (protein (profilin myosin))

Maximal-Common-Subgraph. *Maximal-common-subgraph* finds the largest subgraph that two graphs *u* and *v* have in common. For example, in Table 8

Table 8. Maximal-common-subgraph example

u	*v*
(type (← (protein ApoE)) (→ Apolipoprotein)) (location (← (protein ApoE)) (→ brain))	(agent (→ protein) (← distribute)) (obj (← distribute) (→ lipids)) (type (← protein) (→ Apolipoprotein))

The common subgraph is (type (← (protein ApoE)) (→ Apolipoprotein)) the generic referent (protein) is restricted to the individual referent (protein ApoE).

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 participants in *v* and *u*' have a *maximal common subtype*

Table 9. Match example

u	*v*
(agent (← interfere) (→ flavonoids)) (type (← scaffold) (→ actin)) (obj (← interfere) (→ scaffold))	(agent (← interfere) (→ compound)) (obj (← interfere) (→ assembly))

For example, in Table 9 knowledge lattice operations indicate that (compound) is a supertype of (flavonoids), and (assembly) is a supertype of (scaffold); therefore, the graphs *u* and *v* match.

Projection. Projection 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. For example, in Table 10 adapted from [33],

Table 10. Projection example

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.

4 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.

4.1 Simple Match

Table 11 shows two sentences from two different web pages [34]. These two pages are polysemous at the phrase level.

Table 11. Example of simple match

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.
Query	"interact with snf1"

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 in page 2. This difference is the result of using the Semantic Matcher, which utilizes the Knowledge Lattice and Semantic Matcher operations to compute the relevance of page 2 to the user's query.

Table 12 shows the SP form of each page and the query. The search query in SP form is *(obj (← interact) (→ (protein Snf1)))*. The 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.

Table 12. SP Form of simple match example

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

4.2 Match using synonyms

In the following example (see Table 13) adopted from [35, 36], during the look phase the user query is expanded with "associate", a synonym of "link". And in the find phase semantic matching treats "link" and "associate" as equivalent (i.e. synonyms in KL).

Table 13. Example of match using synonyms

Page 1	***Myosin*** is the most abundant ***protein*** in the human body. Many disorders and health related problems have been ***linked*** to faulty ***myosin***.
Page 2	***Myosin*** associated ***proteins***.
Query	"proteins linked to myosin"

A Google search against these pages with the query "proteins linked to myosin" returns both page 1 and 2 because all the search query keywords appear in page 1, and two of the search query keywords appear in page 2. But an SPS search returns only page 2 because the search query phrase occurs in page 2, but not page 1. In page 1 it is not a protein that is linked to myosin, but disorders and health problems.

The SP forms of the query and pages are shown in Table 14.

Table 14. SP Form of match using synonyms example

Page 1 SP form:	**Page 2 SP form:**
((agent (← be) (→ (protein Myosin))) (manner (← abundant) (→ (quantity most))) (type (← protein) (→ abundant)) (type (← body) (→ human)) (loc (← protein) (→ body))) ((type (← disorder) (→ (quantity many))) (agent (← link) (→ disorder)) (type (← problem) (→ health)) (agent (← link) (→ problem)) (type (← myosin) (→ faulty)) (obj (← link) (→ (protein myosin))))	(agent (← associate) (→ (protein Myosin))) (obj (← associate) (→ (protein (*))))
Query SP form: (agent (← link) (→ protein)) (obj (← link) (→ (protein myosin)))	

No query phrase matches any page 1 first sentence phrases. The page 1 second sentence phrases only match one query phrase, i.e. *(obj (← link) (→ (protein myosin)))* matches *(obj (← link) (→ (protein myosin)))*. However for the page to be accepted, both query phrases must match phrases in the target sentence. The query phrase, *(agent (← link) (→ protein))*, does not match any page 1 second sentence phrase. Therefore, no sentence in page 1 matches the query phrase.

The query phrases match all page 2 phrases. *(agent (← link) (→ protein))* matches (agent (← associate) (→ (protein Myosin))) because the Semantic Matcher recognizes by consulting the KL that "link" and "associate" are equivalent. *(obj (← link) (→ (protein myosin)))* matches (obj (← associate) (→ (protein (*)))) because likewise "link" and "associate" are equivalent.

4.3 Match using subtype / supertype

The following example (see Table 15) adopted from [37, 38] illustrates semantic matching that uses both synonyms and subtype.

Table 15. Example of match using subtype / supertype

Page 1	Flavonoids ***interfere*** with ***actin functions*** in the cytoplasm and the nucleus.
Page 2	***Actin*** filaments were disrupted by Clostridium botulinum C2 toxin.
Query	"Compounds interfering with actin function."

The KL fragment (see Table 16) used by semantic matcher contains specific domain knowledge

Table 16. KL fragment of compound and interfere

Word	Supertype	Subtype	Synonym
compound	( )	(flavonoids cucurbitacin hemslecin gly-coside)	(blend mixture amalgam)
interfere	()	( )	(disrupt obstruct impede block)

A Google search against the pages with the query "compounds interfering with actin function" returns page 1 at the top of the result list because it contains three of the four keywords, and also page 2 further down the result list because it contains one of the keywords. SPS search instead returns only page 1 because the search query phrase matches the page 1 sentence. The query phrase does not match the page 2 sentence, although "interfere" and "disrupt" are synonyms according to the KL, because "Clos-tridium" is not a kind of "compound".

The SP forms of the query and pages are shown in Table 17.

Table 17. SP Form of match using subtype / supertype example

Page 1 SP form:	**Page 2 SP form:**
(agent (← interfere) (→ flavonoids)) (type (← function) (→ actin)) (obj (← interfere) (→ function)) (loc (← function) (→ cytoplasm)) (loc (← function) (→ nucleus))	(type (← filaments) (→ actin)) (obj (← disrupt) (→ filaments)) (type (← toxin) (→ Clostridium)) (type (← toxin) (→ botulinum)) (type (← toxin) (→ C2)) (agent (← disrupt) (→ toxin))
Query SP form: (agent (← interfere) (→ compound)) (type (← function) (→ actin)) (obj (← interfere) (→ function))	

For page 1 the query phrase *(agent (← interfere) (→ compound))* matches *(agent (← interfere) (→ flavonoids))* because according to the KL "flavonoids" is a subtype of "compound". The other query phrases match simply against the page 1 phrases. For page 2 the query phrase *(agent (← interfere) (→ compound))* does not match *(agent (← disrupt) (→ toxin))* because although "interfere" and "disrupt" are synonyms, none of the "toxin" types is an immediate supertype or subtype of "compound".

5 SPS Performance Evaluation

5.1 Evaluation Measures

Two measures, recall and precision, are considered for SPS performance evaluation.

$$recall = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}} \quad (1)$$

$$precision = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}} \quad (2)$$

Recall is the fraction of relevant documents that are retrieved from the collection of all relevant documents. Precision is the fraction of retrieved documents that are relevant. Precision can be viewed as a measure of *exactness*, whereas recall is a measure of *completeness*.

High recall suggests no document has been missed, but also that the results may contain many useless documents (which implies low precision). High precision suggests everything returned is a relevant result, but also that not all relevant items (which implies low recall) could be found.

Every SPS search is a pipeline of two separate searches: a keyword search and a cognitive search. Keyword search uses keywords and an external traditional search engine (e.g. Google, Bing, HotBot, AltaVista, DuckDuckGo, BananaSlug, etc.). Cognitive search uses phrases (i.e. SP forms) and the Semantic Matcher. From here on, we refer to the traditional search engine that is used in conjunction with SPS as the "SPS-pair". For SPS, in the worst-case (i.e. no keyword expansion) recall should be identical to recall of traditional keyword search. In all other cases, recall should be better than recall of traditional keyword search because the traditional search (i.e. 1st step in pipeline) will be conducted with a greater number of keywords that are derived from synonyms and hypo/hypernyms.

Recall, however, cannot be used as an evaluation measure because the size of the document collection (i.e. Internet) is unknown. Therefore, only the precision measure is used in evaluating SPS performance.

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system, i.e., both SPS and its traditional search engine pair. This measure is called precision at *n* or *P@n*. For our purpose n=5.

5.2 Relevance Types

Precision is assessed according to three types of relevance: *Google relevance*, *SPS relevance*, or *human relevance*.

For Google, a page is relevant if any of the keywords (i.e. query sentence words) appear within the page and other pages deemed "authoritative" point to the page. For SPS, a page is relevant if it contains a sentence in which all (or a subset) of the query phrases match the target sentence phrases.

Human relevance is similar to SPS relevance but involves natural language and is performed by a person to assess a test run. The number of natural language query phrases matched in the target sentence determines human relevance. For example,

when n=2 (i.e. the number of query phrases to match) a page is relevant if it contains a sentence which consists of at least two of the query phrases.

5.3 Evaluation Methodology

A single identical query is presented to both SPS and a traditional search engine (e.g. Google, Bing, Yahoo, and Netscape). The top five results of SPS and the comparison search engine are assessed for precision, which is done manually by a person.

One concern is whether the comparison search engine should be different or the same as the SPS-pair? We consider it should be the same because if it is different there will be no way to gauge what weight keyword search and cognitive search respectively contribute to the better performance. If it is different all that could be said is that SPS-Google is better than AltaVista. However, we would not know whether it is the cognitive component (i.e. SPS) or keyword component (i.e. Google) that makes the duo better.

6 Empirical Results

6.1 Experimental Setup

The SPS system shown in Fig. 1 was implemented using Allegro CL Professional Edition 9.0. The two GUIs, "Look-Find" and "Ontology", were implemented only as command line interfaces instead of graphical interfaces. The Knowledge Lattice was populated with 105,898 general knowledge words harvested from online sources [27, 28]. Seven different SPS configuration settings likely to yield the highest precision were selected. Ten queries were created randomly and run using both SPS-Google and Google alone. The top five results (i.e. P@5) of each run of both SPS-Google and Google alone were evaluated for *precision*.

6.2 SPS Configuration Settings

Table 18 shows SPS configuration settings broken into three categories, and the time of their application.

Table 18. SPS Configuration Settings

time	**setting**	**applied during …**
parse	1. preposition discernment 2. phrasal verb 3. multi-word nouns	parse of natural language into SP form
search/look	4. query keyword expansion 5. result set maximum size	search query construction
match/find	6. subtype-supertype depth 7. matched phrases floor	Semantic Matching

Adjustment of these settings causes SPS to produce results of varying/different precision. Note that for consistency in encoding into SP Form, parse time settings once established must be the same for both the query parse and the target pages parse. On the other hand, search time and match time settings, can be changed on the fly because they do not affect the SP Form encoding.

Preposition discernment. One prerequisite to phrase parsing is handling prepositions. In SPS, prepositions map into SP phrase roles (see Table 3 SP form syntax). A single preposition (e.g. to, by, of, in, on, at, as, for, through, between, from, against, into, with, without, before, after, until, during, according, etc.) can express/denote multiple meanings (e.g. location, place, time, purpose, measure, function, etc.) depending on context.

As an example, in the following phrases the preposition "*for*" could denote, depending on context, various meanings (see Table 19)

Table 19. Possible meanings of the "*for*" preposition

Phrase	**Role (specialized)**
"therapies *for* hypertension"	affect
"searching *for* enlightenment"	purpose
"tools *for* making frames"	function
"leaving *for* Boston"	location
"in prison *for* 12 years"	time
"baby crawled *for* 30 yards"	measure
"failed *for* the third time"	series

The "preposition discernment" setting determines whether the phrase role should be general or should be more specialized with the meaning determined by context. By examining the phrase context and the broader meaning of the governing word and its dependent word, a more precise role between two participants would be inferable. For example, the parse of "*a book on careers*" could be encoded either of the two (see Table 20).

Table 20. Example of general / specialized phrase role

(*on* (← book-2) (→ careers-4))	general
(topic (← book-2) (→ careers-4))	specialized

Table 21 shows sample prepositions and the most common category of meaning that each preposition can have.

Table 21. Common meanings of sample prepositions

	to	by	of	in	on	at	as	for	from
time		x	x	x	x	x	x	x	x
location	x	x	x		x	x		x	x
measure		x	x					x	
relation	x		x					x	
medium				x	x				x
cause			x						x
target					x	x			
condition				x		x			
purpose	x							x	
affect	x							x	
scale						x			x
function							x	x	

In addition to the above meanings, other less common preposition meanings are available and some are listed in Table 22.

Table 22. Less common meanings of sample prepositions

to	destination, attached
by	instrument, author, dimension, parameter, deadline, pedigree
of	part, direction, kind, substance
in	enclosure, quality, occupation
on	topic
at	means
as	character
for	occasion
from	source

Phrasal Verbs. A phrasal verb is a multi-word phrase that consists of a verb and some other element, but is treated as a single word with a distinct meaning. For example, “throw-up”, “break-down”, “rolled-out”, “turned-around”, etc. A new parse time setting is introduced, called “Phrasal Verbs”, and if “Phrasal Verbs” is set to “unitized”, then it indicates whether phrasal verbs should be treated as a single distinct unit or as separate words.

Multi-Word Nouns. Multi-word nouns are nouns comprised of multiple words. For example, “Defense Advanced Research Projects Agency”, “Department of Defense”, “George Herbert Walker Bush Jr.”, “Ho Chi Minh City”, “Rio de Janeiro”, etc. Most multi-word nouns, except for person names, are best left indivisible. Note that multi-word nouns are different from noun phrases, which are groups of words that function

as a subject in a sentence. This setting indicates whether multi-word nouns will be treated as a single distinct unit or as separate words.

Query Keyword Expansion. This setting indicates whether each natural language query keyword should be augmented with additional keywords that account for synonymy and the keyword's proper subtypes and proper supertypes. Note that Google limits the number of query keywords to 32. Also, if many (i.e., > 6) keywords are presented or the keywords are in alphabetical order, or have similar spelling, Google retrieves word lists only, and ignores all other pages where those keywords might occur. For example, even when keyword root redundancy is eliminated (e.g. "care" "careful" "caring" "carefulness" => “care”) and keyword order is randomized, Google will still retrieve word lists.

Result Set Maximum Size. This setting indicates the maximum number of result pages a conventional search engine (i.e. Google) should return.

Subtype-Supertype Depth. Subtype-supertype depth indicates the number of KL levels a process may traverse to retrieve a subtype or supertype. One (of many) equality criterion is that two words are equal if they are subtypes or supertypes of each other. This setting allows equality to look beyond the proper subtypes or supertypes to an extended sub/super type.

Matched Phrases Floor.
This setting indicates the minimum number of phrases in the query that must match phrases in the target sentence for the sentence to be considered matched and subsequently, for the page containing the sentence to be a hit. The effect of this setting is to “throttle” precision. Lowering the setting’s value will result in more hits, but a lower precision.

6.3 Query Creation - Random

Queries are constructed using a random approach. Randomness is to assure that operator bias (in favor of SPS) is not introduced in SPS evaluation. The random approach adheres to the following rules:

1. Choose a word at random from the Unix system list of words, /usr/share/dict/words. Ten random numbers (one for each word) that fall in the range 1 and maximum number of words in the list are generated using
   jot -r 10 1 234936
   A word that corresponds to each generated number is selected.
2. Look up the selected word in a dictionary that provides sample sentences containing the word. The following dictionaries, for example, could be used
   http://corpora.uni-leipzig.de

http://sentence.yourdictionary.com/bony

3. Use a sample sentence provided by the dictionary to construct a query. Human involvement is required because composing a query is a human activity.
4. Run the query using both SPS-Google and Google.
5. Evaluate both runs for P@5. Note that using the Leipzig corpora to construct queries provides an additional evaluation advantage. Since the corpora provide the URL of the page that contains the sentence, one knows a priori that at least an actual internet page should contain the query.

The ten queries generated from the above procedure are shown in Table 23.

Table 23. Natural language queries used in trials

	Rand #	**Word**	**Query**
q0	19141	bathing	"Died in a bathing accident in 1978."
q1	79613	greedy	"Greedy corporate capitalists who don't care about the environment."
q2	222998	untangle	"Researchers agree that numbers can be hard to untangle."
q3	68245	fashion	"People decorate their cows everywhere, but we call it a fashion show."
q4	108311	ludicrous	"Talk about anyone appointing the next United States Senator is ludicrous."
q5	88004	huntsman	"The largest huntsman spider has a leg span of 30 centimetres."
q6	171560	sacramental	"Ashes are sacramental but not a sacrament."
q7	232279	woodland	"Green jays inhabit the riverine woodlands."
q8	37622	clogger	"The band and their clogger stopped by the studio."
q9	232464	workhouse	"Contemporary recipes suggest that workhouse gruel was substantial."

Each of the above queries is run using both SPS-Google and Google. SPS-Google has been run multiple times with different choices of SPS configuration settings.

6.4 Trials

Prior to the actual trial runs, pre-trial runs can be carried out to determine desired values for the SPS settings. At maximal configurations, four of the SPS settings can take one of two possible values, one setting a value in the range 1 to 3, and the last setting values ranging from 1 to the number of phrases in the user query. The total number of possible combinations of settings for each query evaluated can be as big as $2^4 * 3 * N$, where N is the number of phrases in the user query. To reduce the complexity, without compromising the trial results, one can choose only those settings that are likely to yield the highest precision. After multiple pre-trial runs we resorted to the following chosen setting values (see Table 24) in our computational analysis:

Table 24. SPS Configuration Settings for Trials

Configuration Setting	**Value**
Preposition Discernment	no (i.e. more ambiguous general role used)
Phrasal Verbs	unitized (i.e. indivisible, treated as unit)
Multi-Word Nouns	unitized except person names
Query Keyword Expansion	no (because expansion may result in word lists)
Result Set Maximum Size	40 pages
Subtype-Supertype Depth	level 1 only (i.e. proper subtype or supertype)
Matched Phrases Floor	varied from user query number of phrases to 1

All trials have been run under these settings.

Table 25. SPS Trials Result Data

		Google										**SPS**				
		rank					**count**					**count**				
	N	**n**	**n-1**	**n-2**	**n-3**	**n-4**	**n**	**n-1**	**n-2**	**n-3**	**n-4**	**n**	**n-1**	**n-2**	**n-3**	**n-4**
q0	3		10	23				1					3	20		
q1	5		33	5	5	5		1	1	1	1		1	4	16	24
q2	4		12	2	2			1	2	2			1	4	10	
q3	9				26	24									2	4
q4	5		5	5	5	2		1	1	1	4		1	1	10	18
q5	7	36	36	6	1	1				3	4	1	2	7	21	30
q6	2	24	3					1				2	6			
q7	4	9	1	1					1	3			1	4	16	
q8	5					1					2					5
q9	5		1	1	1	1		3	3	3	3		12	12	13	26

Table 25 shows the results of the runs of the ten queries (i.e. q0 – q9). The column headings are meant to represent their respective notions given below:

- *N* is the number of phrases in the query.
- *rank* is the position, from the top of the result list, of the first page hit. A hit is a page, human relevant to the query (i.e. page contains at least one sentence that matched the query). Rank is noted only for Google, because a relevant Google result page might appear anywhere within the set of result pages. SPS returns only pages that are hits; therefore, SPS rank will always be one.
- *n* under *rank* is the position, within the result set, of the first hit from the top where all N phrases match.
- *n-1* under *rank* is the position, within the result set, of the first hit from the top where all n-1 phrases match. Similarly, for all the other "*n-*" columns under *rank.*
- *count* is the number of hits within the first 5 pages from the top.

- *n* under *count* is the number of hits within the first 5 pages from the top where all N phrases match.
- *n-1* under *count* is the number of hits within the first 5 pages from the top where all n-1 phrases match. Similarly, for all the other "*n-*" columns under *count*.

Trials – Comparative Results. Figures 3-9 show a graphical representation of **Table 25** data.

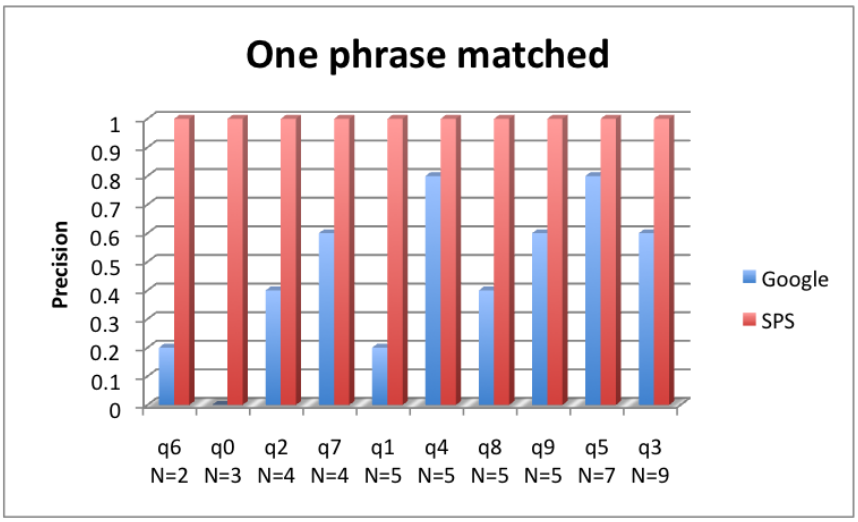

Fig. 3. P@5 One phrase matched

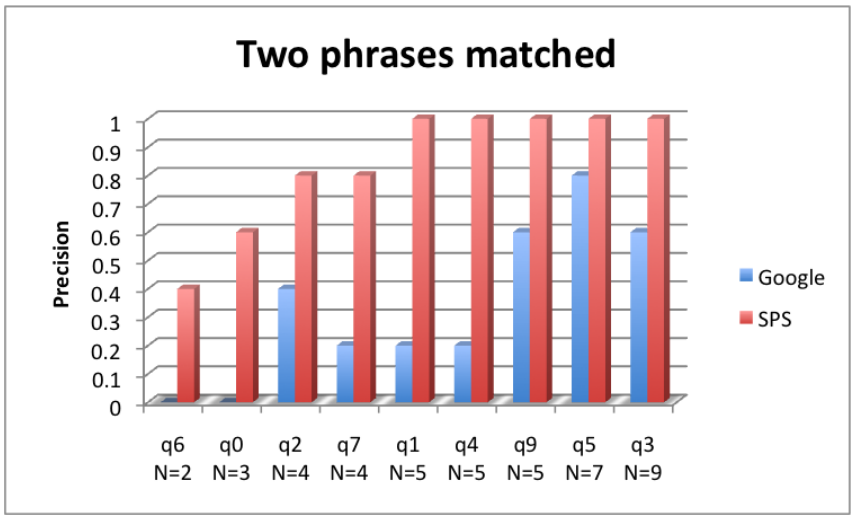

Fig. 4. P@5 Two phrases matched.

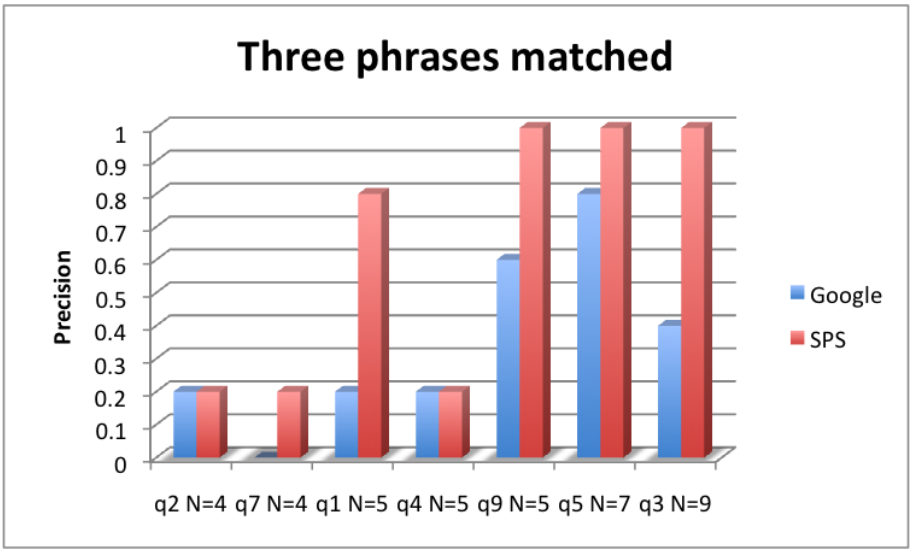

Fig. 5. P@5 Three phrases matched.

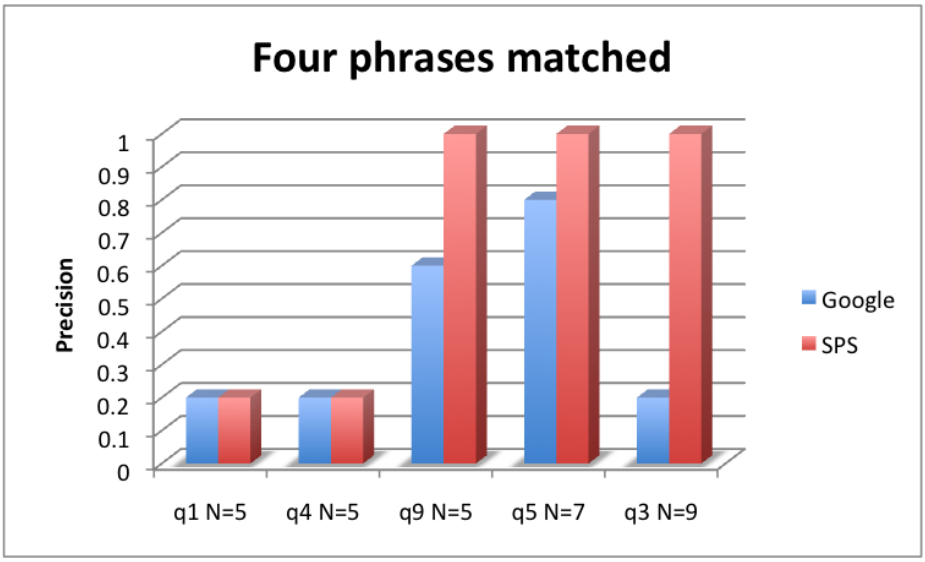

Fig. 6. P@5 Four phrases matched.

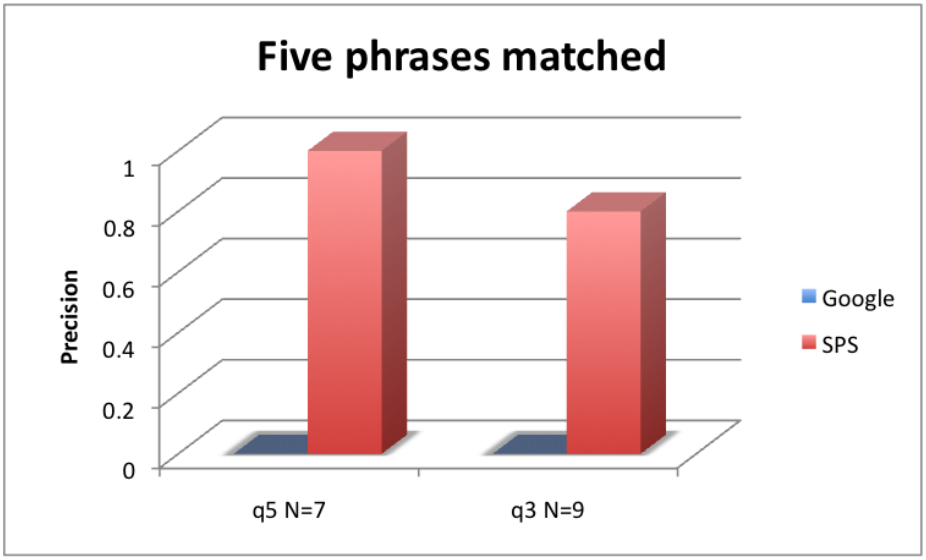

Fig. 7. P@5 Five phrases matched.

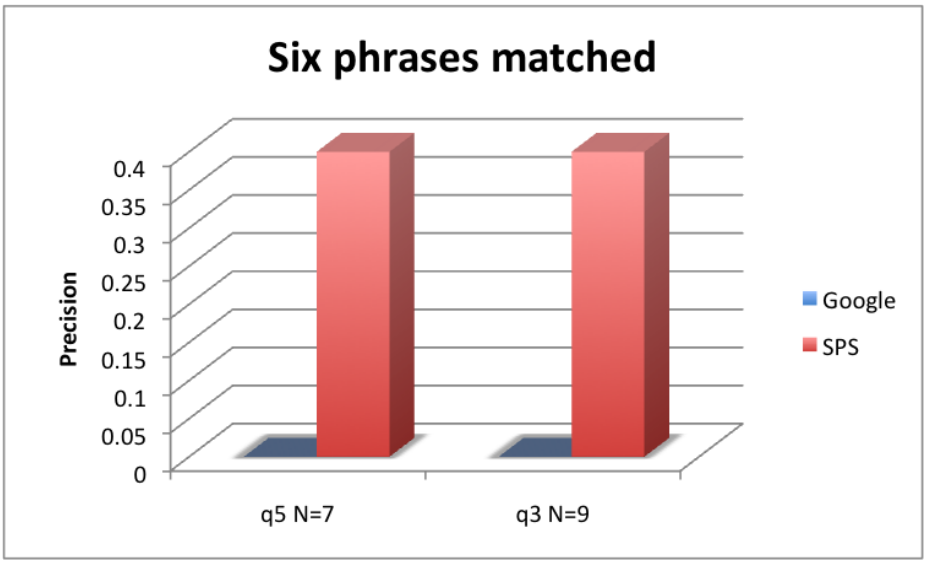

Fig. 8. P@5 Six phrases matched.

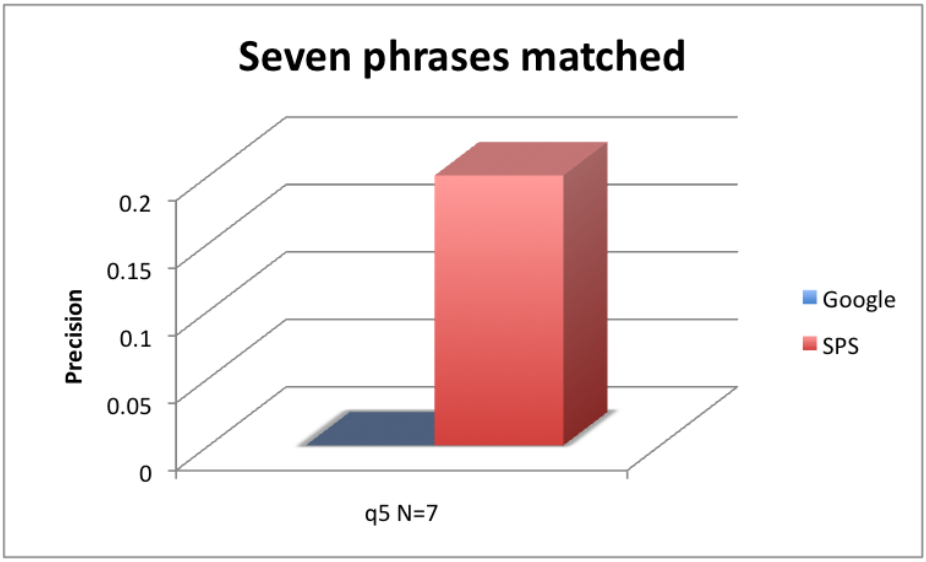

Fig. 9. P@5 Seven phrases matched.

When n=1 (**Fig. 3**) Google and SPS are on an almost equal footing, their match atomic units are very similar, a *keyword* in Google and a single *phrase* in SPS. However, SPS still has an advantage over Google.

With two phrases (**Fig. 4**), the precision of most queries is reduced. The reduction is most noticeable in queries that have approximately two phrases. Google precision is the most reduced in q7 and q4; q6 is dropped. For both Google and SPS, q8 is dropped.

When n=3 (**Fig. 5**), matches are even more restricted. For both SPS and Google, q6 is not considered and q0 is dropped. For SPS, q4 precision is substantially reduced. For Google, q3 precision is substantially reduced, and q7 is dropped.

When n=4 (**Fig. 6**), q2 and q7 which are made up of four phrases, are dropped. SPS q1 and Google q3 precisions are reduced.

When n > 4 (Figures 7-9), only SPS produces P@5, and the precision decreases considerably. With each increase in the number of phrases matched, precision is about halved.

Trials – Hit Dispersion. For every trial Google returns 40 pages it considers relevant, ordered from most to least relevant. SPS examines the Google result set, and selects from the set those pages that are *SPS relevant*. Note that the pages examined by SPS are exactly the Google result set. This is because keywords, for the reasons cited in Section 6.2.4, are not expanded. In effect, SPS sifts through the Google result set to find the *human relevant* pages. The hit dispersion figures below show the dispersion of human relevant pages in the Google result set.

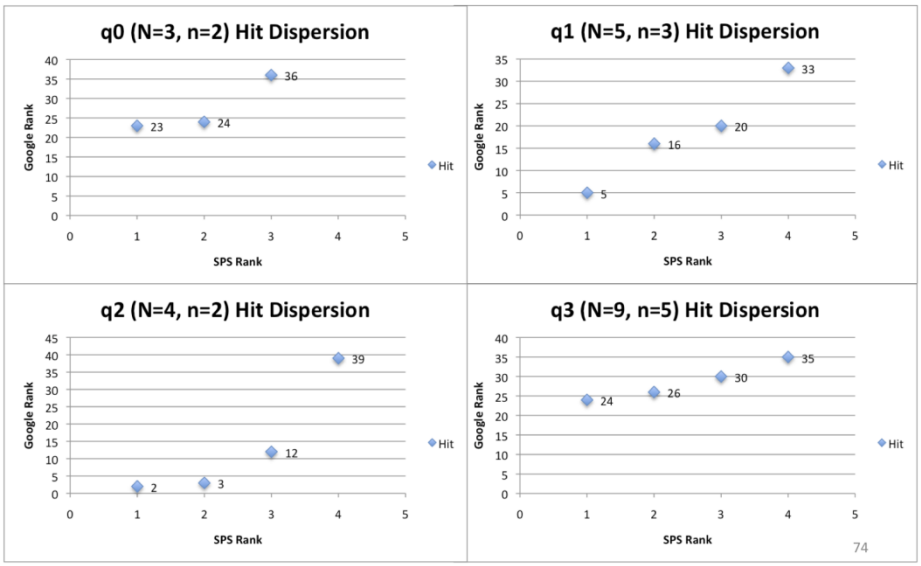

Fig. 10. Hit Dispersion (q0-q3)

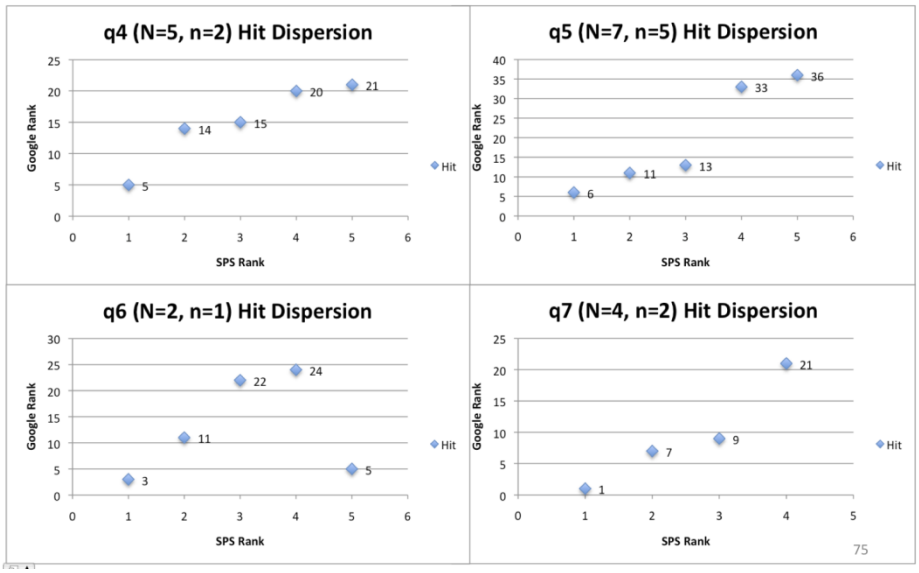

Fig. 11. Hit Dispersion (q4-q7)

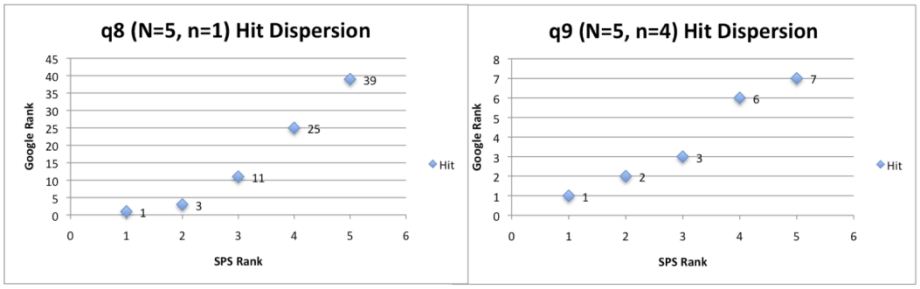

Fig. 12. Hit Dispersion (q8-q9)

7 Discussion

Table 25 and the graphs in Sec 6.4.1 and 6.4.2 show, for each query, the precision comparison between Google and SPS-Google, and the distribution of page hits in each method. From this data we can make the following observations.

Matching all query phrases is too restrictive. When n=N only two (q5, q6) of the ten queries result in SPS hits (see **Table 25**). Out of 40 pages, SPS finds for q5 only one human relevant page and for q6 only two human relevant pages. At P@5 and n=N, Google does not retrieve any human relevant pages for any query.

SPS precision increases as the Matched Phrases Floor (see Sec 6.2.7) decreases. For example, q6 which consists of two phrases (N=2), goes from a precision of 0.4 when n=2 (**Fig. 4**) to 1.0 when n=1 (**Fig. 3**). Matched Phrases Floor is applicable to Google only for human relevance.

The variance in the hit count between matching all query phrases and matching a subset of the query phrases is small for Google but quite large for SPS. For example, the variance between an n-1 count and n-4 count for Google is small; in fact the largest variance is 3 in q4 (see **Table 25**). In SPS the variance is much greater, the largest variance is 29 in q5 (see **Table 25**).

These differences in how hits are dispersed in the result sets of the two approaches are expected because in our opinion the Google approach is “sloppier” than the SPS approach. Here by "sloppy" we mean that in Google a page is a hit if any of the keywords appear anywhere in the page even if by happenstance those keywords are unrelated to the page content (e.g. keyword contained in an advertisement). See Hit Dispersion Figures 10-12.

SPS precision is always higher than Google precision, even when n=1 (i.e. match only one phrase). When n=1 Google and SPS are on an almost equal footing, and thus their match atomic units are very similar, i.e., a keyword in Google and a single phrase in SPS. See **Fig. 3**. However, SPS still has an advantage over Google. The SPS advantage is due to the fact that the SPS basic unit of matching, the *phrase*, is more constrained than the Google basic unit of matching, the *keyword*. A phrase is made up of multiple keywords inter-related by a role. Because the words are inter-related their scope is restricted. Moreover, the SPS match target, i.e., the *sentence*, is more limited than the Google match target, the *document*. In Google, the same words might have a

much broader scope (for example, two words might be far apart within the document as they are searched separately). In SPS, the two words will have a narrower scope (i.e. the two words will be, not just within the same document, but within the same sentence) because they are inter-linked by a role. This difference in scope (i.e., a sentence vs. a document) explains the higher precision for SPS. Consequently, SPS should most of the time produce higher precision/relevance and such observation is supported by the comparisons illustrated in Figures 3-9.

8 Conclusion and Future Work

This work demonstrates that phrase search can produce Web search 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 constitute some meaning. The inter-relation expresses grammatical and semantic information, and it 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. For the future work, we plan to improve SPS by tackling more advanced topics such as knowledge-based stemming, KL curation and verification, and NL parsing which still can be further improved. We also plan to examine use of crawling and phrase indexing, broken word repair, and sentence recognition. Each of these advanced topics is briefly explained below.

Knowledge-based stemming. All words that enter SPS are stemmed. Stemming ensures that all variants of a word are represented by a single word. For example, "*increase*", "*increased*", "*increases*", and "*increasing*", all stem to "*increase*". SPS uses the spelling based Porter stemming algorithm [39]. Our experience suggests that spelling based stemming can causes two anomalies.

1. Words with dissimilar meaning might produce the same stem. For example, "*depart*" and "*department*" stem to "*depart*". One means to go or to be deceased, whereas the other refers to a division of an organization. "*Hers*" and "*herring*" stem to "*her*". One is a possessive pronoun, and the other is a fish. Consequently, reverting a stem to raw words might produce words that are not semantically related. For example, "*depart*" will revert to both "*department*" and "*departed*".
2. Words with the same meaning might produce dissimilar stems. For example, "*spin*" and "*spun*" stem to themselves, or "*airplane*" and "*plane*" stem to "*airplan*" and "*plane*" respectively. An aliasing mechanism handles dissimilar stems that have the same meaning.

At the moment, these anomalies in stemming are not detrimental to SPS functionality because dissimilar words that occupy the same node in the KL generate different threads of meaning; however, some of these meaning threads affect performance during SPS processing because they reach dead-ends. Therefore, a spelling based stemmer should be replaced with a semantically guided knowledge-based stemmer.

KL curation and verification. The KL currently consists of 105,898 words supplied from three sources: harvested from Internet dictionary [27] (98%), auto-learned, or provided by a person (863 words). The KL is not concerned with word meaning but with variation in meaning between words. KL edges are to represent this variation in

meaning. Because the initial automatic build consisted of large number of words, every one of these edges could not be checked. However, we can assume that any word harvested from a dictionary or provided by a person is correctly placed in the lattice. Therefore, the only words that are possibly mis-positioned are auto-learn words (see Section 3.4). Auto-learn words can be curated and verified by a person (or crowd); non auto-learn words can be checked by random sampling followed by verification.

NL Parser. The Stanford Dependency (SD) parser [24] is mostly concerned with grammatical elements (i.e. passive, active, infinitival, etc.) that denote fine distinctions in meaning. For SPS, these subtle distinctions are not important because internally all derivatives of the same concept are represented by a single word. Moreover, computing these fine grammatical distinctions slows down both the SD parser and the translation of SD parser output to SP Form. NL parsing is at the core of SPS. Consequently, an NL parser needs to be developed that can be tightly integrated with the KL to leverage both "world knowledge" and grammatical information about words. Such a parser would translate natural language to SP form in one step and ideally, at the same time, perform the auto-learn function.

SPS Crawler & Phrase Indexing. SPS depends on Google to look for relevant information from the Internet. The next step in SPS evolution is to replace Google with a crawler that indexes phrases instead of words. In conjunction with the NL parser described above such a crawler would, in addition to indexing phrases, also auto-learn (i.e., contextually infer the meaning of new words) from the pages that it crawls (i.e. reads).

Broken word repair. Pages that include reader feedback forums or that have been OCRed often contain many mis-spelled words (e.g. "sceintist", "asume", "work done oy prisoners", "outeide work", "com- pulsory", etc.). Reader feedback forums also contain strangely spelled user names (e.g. "MissPoo", "albatroll"), which won't exist in any dictionary or encyclopedia. SPS needs to include a facility that determines whether an unknown word should be ignored (e.g. "albatroll"), repaired (e.g. "outeide"), or looked up (e.g. a word not in the KL).

Sentence recognition. The parser operates on individual sentences, which need to be extracted from the page collection of sentences. Determining where a sentence starts and ends is a difficult problem. Currently, a sentence starts with a capital letter and ends with period, exclamation, or question mark. However, some sentences are only partially recognized by this definition. SPS needs to incorporate a more sophisticated way of recognizing sentences.

9 References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. In: Proceedings of the Seventh International Conference on World Wide Web 7 (WWW7), Philip H. Enslow, Jr. and Allen Ellis (Eds.). Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, pp. 107-117 (1998)
2. Leone, J: "A Phrase-based Ontology Enabled Semantic Processing System for Web Search", Ph.D thesis, University of Connecticut, (2013) (To be published)
3. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)
4. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema (10 February 2004). http://www.w3.org/TR/rdf-schema/ (Retrieved August 2013)

5. McGuinness, D., van Harmelen, F.: OWL Web Ontology Language Overview (10 Feb 2004). http://www.w3.org/TR/2004/REC-owl-features-20040210/ (Retrieved August 2013)
6. Klyne, G., Carroll, J.: Resource description framework (RDF): Concepts and abstract syntax (February 2004). http://www.w3.org/TR/2004/REC-rdf-concepts-20040210 (Retrieved August 2013)
7. Ding, L., Finin, T., Joshi, A., Pan, R., Scott Cost, R., Peng, Y., Reddivari, P., Doshi, V., Joel Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM '04). ACM, New York, NY, pp. 652-659 (2004)
8. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: Proceedings of the 17th International Conference on World Wide Web (WWW '08). ACM, New York, NY, USA, pp. 1101-1102 (2008)
9. Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A., Stefan Decker, S.: Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. Web Semantics 9(4) 365-401 (December 2011)
10. d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E.: Watson: A Gateway for Next Generation Semantic Web Applications. Poster, ISWC (2007)
11. Dietze, H., Schroeder, M.: GoWeb: a semantic search engine for the life science web. BMC Bioinformatics 10(Suppl 10):S7 (2009). http://www.biomedcentral.com/1471-2105/10/S10/S7 (Retrieved August 2013)
12. Leone, J., Shin, D.: Knowledge-Based Visualization of Textual Information Applied in Biomedical Text Mining. The Seventh International Multi-Conference on Computing in the Global Information Technology (ICCGI 2012). 24-29 June 2012, Venice, Italy.
13. RDFa (Resource Description Framework in Attributes) is an extension to HTML5. http://rdfa.info (Retrieved August 2013)
14. Adida, B., Herman, I., Sporny, M., Birbeck, M.: RDFa 1.1 Primer (June 2012). http://www.w3.org/TR/xhtml-rdfa-primer (Retrieved August 2013).
15. Hickson, I.: HTML Microdata. http://dev.w3.org/html5/md/Overview.html (Retrieved August 2013).
16. Metadata embedded in Web pages that can be exploited by search engines. http://microformats.org (Retrieved August 2013).
17. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (21 May 2004). http://www.w3.org/Submission/SWRL/ (Retrieved August 2013)
18. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
19. Horrocks, I.: Ontologies and the semantic web. Communications of the ACM, 51(12), 58-67 (2008)
20. Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., Sherlock, G.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics* 25(1), 25-29 (01 May 2000)
21. Medical Subject Headings (MeSH). http://www.ncbi.nlm.nih.gov/mesh (Retrieved August 2013).
22. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, (1984)

23. Sowa, J., Dietz, D.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks/Cole Pub Co, (1999)
24. The Stanford Dependencies manual. http://www-nlp.stanford.edu/software/stanford-dependencies.shtml (Retrieved August 2013)
25. Genesereth, M., Ketchpel, S.: Software Agents. Communications of the ACM 37(7), 48-53 (1994)
26. Lenat, D.: CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33-38 (1995)
27. A hyperlinked dictionary. http://www.hyperdic.net (Retrieved August 2013)
28. The Wikipedia web-based encyclopedia. http://en.wikipedia.org (Retrieved August 2013)
29. Unified Medical Language System (UMLS). http://www.nlm.nih.gov/research/umls (Retrieved August 2013)
30. Haas, N., Hendrix, G.: An approach to acquiring and applying knowledge. In: Proceedings of AAAI, pp. 235-239 (1980)
31. Haas, N., Hendrix, G.: Learning by Being Told: Acquiring Knowledge for Information Management. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (Eds.), Machine Learning: An Artificial Intelligence Approach. Morgan Kaufmann Publishers, Inc., vol. 1, Chapter 13, (1983)
32. Niagara Falls Thunder Alley magazine. http://www.niagarafrontier.com/accident.html (Retrieved August 2013)
33. Fargues, J., et al.: Conceptual graphs for semantics and knowledge processing. IBM Journal of Research and Development 30(1), (January 1986)
34. Information Hyperlinked Over Proteins (iHOP) Scientific Literature Project. http://www.ihop-net.org/UniPub/iHOP/gs/32484.html (Retrieved August 2013)
35. Myosin Structure. http://www.cs.stedwards.edu/chem/Chemistry/CHEM43/CHEM43/Myosin/STRUCT~1.HTM (Retrieved August 2013)
36. Madame Curie Bioscience Database. http://www.landesbioscience.com/curie/chapter/2152/ (Retrieved August 2013)
37. American Society for Cell Biology (ASCB) Molecular Biology of the Cell (MBoC). http://www.molbiolcell.org/cgi/content/abstract/5/11/1199 (Retrieved August 2013)
38. ScienceDirect. http://www.sciencedirect.com/ (Retrieved August 2013)
39. The Porter Stemming Algorithm. http://www.tartarus.org/~martin/PorterStemmer (Retrieved August 2013)