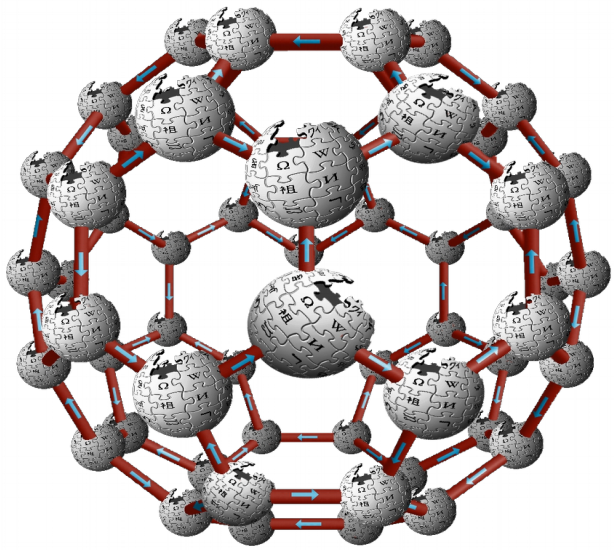
SemantiGrid Knowledge Base - SGSKB™

Chicago Technologies Incorporated

Abstract

Text analytics and its more demanding relative – Natural Language Processing (NLP) - are increasingly becoming a focus of the business and our society. Where Business Intelligence has mostly focused on gaining business value from operational structured data within data warehouses, there is now the desire to mine the vast ocean of data that is available within the larger web along with a business internally collected data. There is also the need to analyze sentiment and determine how customers are viewing services and products.

Unlike like document text search, NLP, semantic analysis and sentiment analysis require accurate analysis of human speech. Existing tools and approaches are proving to be inadequate for the kind of processing and functionality required. It is for this reason that SGSKB™ was created. The foundational work for SGSKB™ is based on the Ph.D. research thesis in "question answering" *(Mlynarczyk, Stanley J., (2009). Investigations of mechanisms to improve question answering. DePaul University (2009))* where existing capabilities in semantic analysis were shown to be inadequate, both in terms of functionality and performance.



Why A New Tool

While there are tools available within the text analytics and NLP space, many of these focus on function as opposed to performance. SGSKB™ was designed to provide advanced NLP functionality yet with performance being a key priority. SGSKB™ performance allows it to be used both in a low latency streaming implementation and for analytics requiring processing of very large data volumes. ***Many functions are "sub-microsecond"***. This allows a ten node cluster to easily provide a billion lookups/second and is linearly scalable to the number of nodes. Both LINUX command line shell and high performance API libraries (shared and archive) are available along with Java JNI support.

Why Semantic Analysis?

Advanced text analytics requires a variety of mechanisms often working in concert to achieve the level of accuracy often sought within business solutions. Basic approaches such as word frequency produce reasonable results but often are insufficient in the accuracy being sought. This is because the mechanism of word match is missing two things – consideration of synonymy and semantic context. The English language is very rich with many words and combination of words that can often produce similar meanings. It is easy to achieve a measurable level of success using simple word match but exceedingly difficult to increase accuracy beyond this base level.

The difficulty manifests itself in the amount of compute power required to just consider synonymy This however is the easier problem to solve given the abundance of compute power available. The computation problem however is exponential once we deviate from simple word match and can be costly to solve.

A more difficult challenge is in the area of semantics. The human mind can discern the different meanings of words within a sentence because of real-world knowledge. A computer unfortunately has no context for discernment of human speech. Consider the sentence "Had the Babe not drank himself silly the night before, the contest would have ended much differently given a few home runs". For most Americans, the above sentence is perfectly clear. Computationally, this same sentence is undecipherable without advanced analytics. We need to understand that "the Babe" refers to a specific baseball player. This is no small task given that the word "babe" can also mean "baby" or "voluptuous woman". The phrase "drank himself silly" is equally problematic and must be interpreted to mean "drunk" or "inebriated". "Contest" in this case must be equated to baseball game. The term "home run must be recognized as a named entity". Lastly, the Babe must be correlated with an actual player name that might have appeared in a prior sentence.

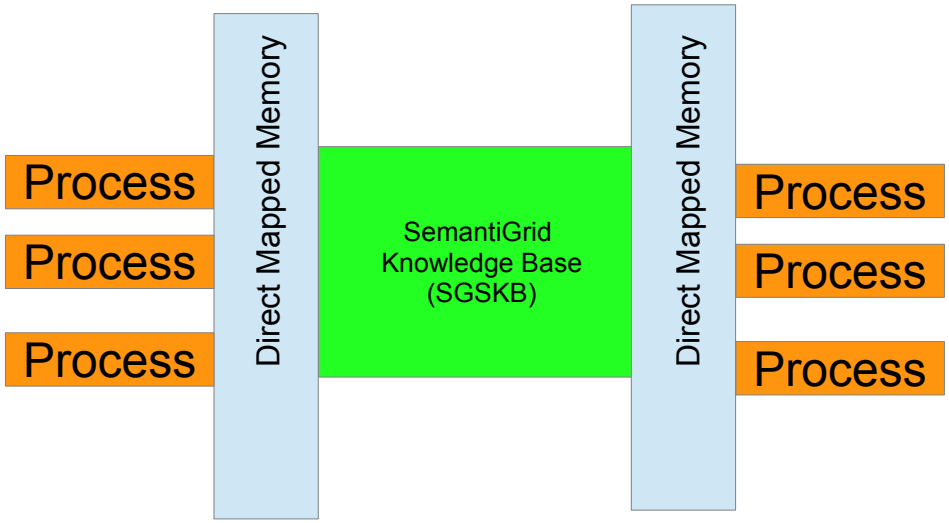
How can all of the above be accomplished? The SGSKB™ can provide help in this case. Via phrase and named entity resolution, the SGSKB™ can resolve "drank himself silly" to be "been drunk" and "home run" to be the baseball sense "home-run". The SGSKB™ can also recognize "the Babe" as being "Babe Ruth" via "name entity resolution". A more difficult task is determining that "contest" is referring to a baseball game. This can be accomplished via semantic distance calculations between all of the words in the sentence and the word "contest" to arrive at a synonym of the term "baseball game".

Human speech is ambiguous and to adequately disambiguate the meaning of text, there is a need to go beyond the simple word matching algorithms such as can be found in Lucene or Elastic Search (which

is based on Lucene). SGSKB™ has been created to provide the semantic methods that allow a much deeper analysis of text and yield increased accuracy in text matching.

SemantiGrid Knowledge Base (SGSKB™) – The Text Analytics Engine

The SemantiGrid Knowledge Base (SGSKB™) is an in-memory N-Dimensional repository of information. Integrated within its structures is the Webster Dictionary, synonym relationships and word relationship (semantic similarity, senses, ...). The repository is hierarchically structured but augmented with additional relationships to form the N-Dimensional features. The memory resident repository allows multi-process/multi-user direct access. This approach provides extremely fast access, search and relationship analysis.



The above shows the direct mapped memory area with multiple client program attachments possible. The importance of this is approach become clear where the frequency of word resolution is large. Even the need to do system calls during lookup functions becomes a performance issue. Not only are all SGSKB™ functions minimally depended on system calls, but the access library is "direct memory access" to the SGSKB™ knowledge repository. ***This means that SGSKB™ traversals are sub-microsecond in a majority of accesses.***

The end goal with the SGSKB™ is to make semantic analysis possible. Each node within a cluster will have its own active copy of the SGSKB™. This ensures optimal performance regardless of the computing platform and makes the SGSKB™ an attractive option for all text analytic applications. The SGSKB™ is built into SemantiGrid but may be deployed on Hadoop and all major LINUX/UNIX-based OLTP (Oracle, DB2, MySQL, PostGreSQL, …) and EDW databases (Oracle, DB2, Netezza,

Teradata, ...)

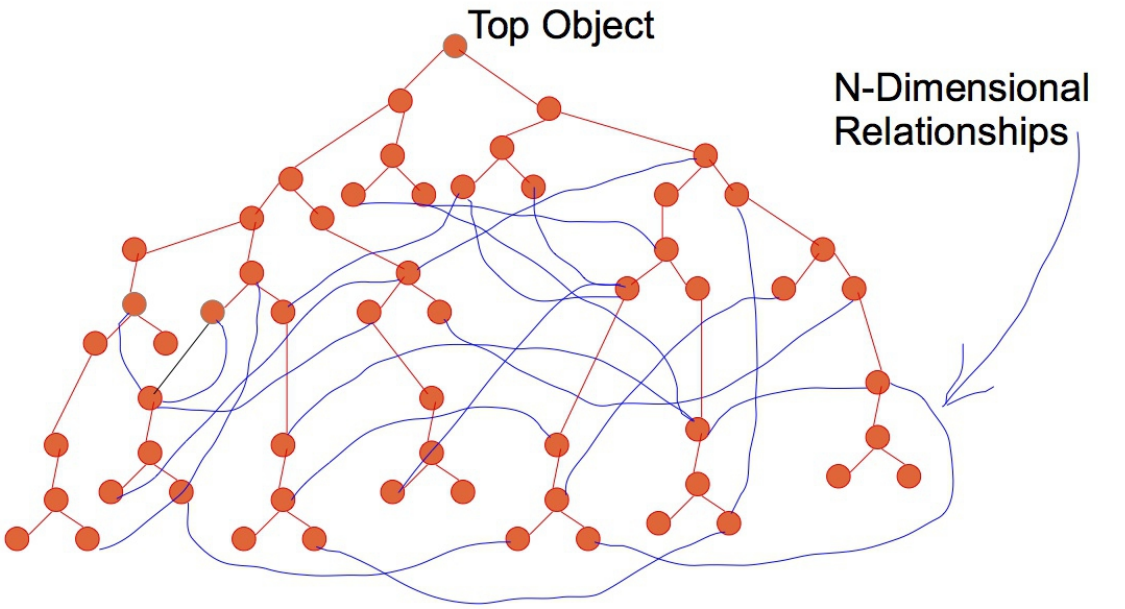
Why have a knowledge base when there are approaches using a database for operations such as synonym lookups or computation of word sense or word distance? While it is true that one can take a database/SQL approach for this task, a major consideration will be scale. If the document store to be scanned is relatively small, a database approach can perform adequately. It is when the document store is large(r), the difference in approach becomes apparent.

Consider a database approach where there is a need for a simple lookup of a synonym. The operation will require a database connection (this might be a one-time occurrence for many accesses), a SQL operation and potentially a join depending on table structure. A SQL database's data is usually controlled by a (server) and a client must interact with the server over a network (or at least an on-system socket).

Contrast the above with a client that has the SGSKB™ at its disposal. The client will call a function that will simply perform a series of memory accesses to find the desired information. There is at least two orders of magnitude performance improvement in use of the the SGSKB™. While pure performance may not be of significant value for small data domains, this becomes a critical factor as the user load increases and especially as the document store becomes large. The SGSKB™ can potentially support millions of accesses in the time a single SQL operation might take for one lookup.

Below is a graphical representation of the SGSKB™ structure. It is both a hierarchical tree and an N-dimensional set of relationships.

Ordered Tree



Word/Object/Phrase

Parent/Child

Relationship (various)/Class/SuperClass

It must be emphasized that the SGSKB™ approach becomes a critical component of any deep semantic or even shallow semantics endeavor on large datasets. The reason for this is that analysis of text requires not only considerations of synonymy but also word sense to produce high accuracy. Implementation of these concepts is often an exponentially increasing workload due to the rapidly expanding number of words/sense that must be evaluated. This generally must be done both for the text being search (as well) as the search terms or document. It is easily well beyond (X**2) * (Y**2) where X is the number of words in the search terms/document and Y is the number of words in the document store.

For this reason, the SGSKB™ needs to be highly efficient to have any hope of providing reasonable performance. Details of the SGSKB™ implementation include:

- Multi-tenant (shared copy of the knowledge store)
- Ultra fast accesses (usually sub microsecond lookups)
- Tree structure augmented with N-dimensional relationships
- Currently there are
  - Over 190,000 object nodes
  - More than 830,000 relationships
  - More than 1,000,000 total entities
- Support for adding additional data domains such as medical or scientific.
- Support for additional languages (other than English).
- Support for phrases (comes with over 3,000 phrases)
- Support for ISA relationships
- Part of speech support (full dictionary)
- Fuzzy match (ultra fast because of the SGSKB™)
- Antonyms/holonyms/meronyms/synonyms/hyponyms
- Sentiment analysis functions for both text and text files
  - Ability to add/modify sentiment keys
- Domain classification of text and text files
  - Ability to add/modify domain classes
- Custom relationships
- SGSKB™ relationships can be added either via user interface, programing API or batch load
- SQL-like command line user and administration shell
- Built-in performance statistic functions
- Self synchronizing knowledge repository across a cluster
- Backup/restore/versioning capability
- LINUX C/C++ programming API support via shared and archive libraries
- JAVA support via JNI

In contrast to the SGSKB™, approaches that use file IO are orders of magnitude less efficient. For small workloads or very shallow semantics, this may suffice. For efficient processing of deeper semantics or large scale data, file IO is a very significant roadblock to a workable solution.

Lastly, the SGSKB™ supports the ability to input new relationships. This is accomplished in one of

two ways; an interactive shell that supports a SQL-like command set or via a file of isa-like relationship language statements. With this capability, the KB is extensible well beyond its initially populated data.

SGSKB™ Performance

Included within the SGSKB™ are a set of performance measurement tools to allow the gathering of performance statistics. These provide a way to measure performance across different hardware platforms. As an example, a snapshot of performance was gathered on a modestly configured 1.8Ghz Intel-based system with the following results:

- 63 (full) SGSKB™ scans within a second for a total ~12 Million node visits/sec. This demonstrates the pure traversal power of the SGSKB™. Each node within the SGSKB™ is visited using the standard search/traversal method used for all SGSKB™ access.
- ~938,700 synonym lookups/second with a total of 1.5 million synonyms found. The test takes each word within the SGSKB™ and performs a lookup for (all) of its synonyms.
- Word distance computations at an average rate of 17,384/sec. This test takes each word within the SGSKB™ and computes the semantic distance to every other word in the SGSKB™. It is N**2 where N is the number of words in the SGSKB™. This is a most computationally intensive task because of the need to build word distance graphs for every word and involves much back-tracking within the ontology structure.
- Note that a multi-core system will scale ~ linearly by the number of CPUs

The above are figures for a single CPU node with a modest clock speed. To put forth further perspective, a single node with an 8 core CPU will be expected to generate (8 execution threads) ~100,000,000 node visits, ~139,000 word distance computations and ~7.5 million synonym lookups per second.

The following table provides a tabular view of a single single-core CPU, 8 core CPU and a 10 node cluster. The below figures are modest in that an 8 core enterprise level CPU would be of a higher clock frequency and equipped with faster memory.

	1 CPU Core	8 Core (single node)	10 node cluster (80 cores)
Object/Word Full Scans	12 Million Words/sec	100 Million Words/sec	1 Billion Words/sec
Synonyms	938,700/sec	8 Million/sec	80 Million/sec
Word Distance	17,385/sec	140,000/sec	1.4 Million/sec

The above figures are modest yet they convey a sense of what is possible. This can be contrasted with a SQL implementation that often is used for such operations. There is a multiple orders of magnitude difference between what can be achieved by utilizing the SGSKB™ over what is possible with a database. Even SQL databases that have been engineered to provide very good/scaled performance for a given SQL operation are (comparatively) inefficient in handling high volumes of SQL operations. The SGSKB™ on the other hand has been designed for performance that can truly make NLP possible.

Business Production Focus

SGSKB™ has been designed with a business production environment as a requirement. While a clustered environment will have a local copy on each node, these copies will communicate with each other to ensure that any updates to the knowledge base will be propagated to all instances. Once the SGSKB™ has been launched, it requires no further administration to be usable within a solution and there is no need to monitor any agents or services.

There is no cluster-wide configuration required. If a new node is added to a cluster, the SGSKB™ is simply started and it will begin communicating with other nodes' instances to get into a synchronized state. A startup script is provided that integrates with a LINUX system's init.d service startup mechanism to provide hands-off, maintenance-free peace of mind.

Data Protection

SGSKB™ has been designed with data protection as a key feature. This is important because a newly installed instance is assumed to be dynamic in the sense that it will be customized with new relationships. These new relationships must be protected and the SGSKB™ provides both backup/restore and version support for the knowledge repository. As important as backup and restore, a distributed environment will have potentially many copies of SGSKB™ running independently. All of these instances must have the information within their knowledge bases. For this reason, all SGSKB™ instances automatically synchronize their data repositories and provide a single, consistent knowledge base across all participant nodes.

A user or administrator can execute the command line shell to verify the operational state of the SGSKB™ knowledge store. This will provide a rich assortment of statistics about the current state of the SGSKB™ , including a traversal and consistency check of the entire knowledge base. The same command line shell will permit an administrator to reinitialize the knowledge base from either the most recent backup or any prior version.

Below is an example of the kind of statistics available*:

Total object Count=	190,500
Phrase Count=	3,059
Synonym Count=	749,946
Phrase Synonym Count=	4,447
Meronym Count=	22,187
Holonym Count=	22,187
Homonym Count=	877
Hyponym Count=	8,577
Antonym Count=	7,979
Relation count=	831,315
Domain_synset count=	6,602
Member_of_Domain count=	9,390

Total SGSKB Count= 1,024,874
Shared Memory Allocated= 318,767,104
Shared Memory in Use= 304,263,700
Shared Memory free= 14,503,404

* Counts will vary depending on any additions/changes to the knowledge base.

In addition to the above health check, the SGSKB™ maintains a log of its operational state. Any errors encountered in servicing user requests are also logged.

Availability

Once the SGSKB™ is started and successfully initializes, there is peace of mind that it will be servicing user requests as long as the node and the operating system for a given node are operational. There is no need to monitor the SGSKB™ operational state. All access to the SGSKB™ repository is via a command line shell or programmatic API.

Programmatic and User Data Access

The most performant way to access the SGSKB™ knowledge base is via the programming API. The API provides a rich assortment of functions that ensure both integrity and the highest level of performance. A sample of provided functions include:

Administration

- backup
- restore
- dump (print all information about an object)
- show (backup versions available)
- clear (command line instruction to clear the repository)
- performance (run performance check and provide statistics)

Client Access

- insert (object/attribute/...)
- select
- update
- delete
- describe (print the structure of table)
- show (objects/tables/structures)
- distance (semantic distance between two words)
- performance (run performance check and provide statistics)

Hadoop Integration

The SGSKB™ can be installed on a Hadoop cluster to provide very high performant NLP functions. The two mechanisms are via the command line shell described above or via the included shared/archive libraries. For example, here is a (partial) list of the functions available:

SGSKB_Init()
SGSKB_Backeup()
SGSKB_Restore()
SGSKB_SetDefaultDelimiter()
SGSKB_ClassifyDomain()
SGSKB_ClassifyDomainFile()
SGSKB_GetAntonyms()
SGSKB_GetDescription()
SGSKB_GetFuzzyMatches()
SGSKB_GetMeronyms()
SGSKB_GetHolonyms()
SGSKB_GetHomonyms()
SGSKB_GetPhrasesByWord()
SGSKB_GetPhraseSyn()
SGSKB_GetPhraseSyns()
SGSKB_GetSynonyms()
SGSKB_GetDescription()
SGSKB_GetSemanticDistance()
SGSKB_GetSynSemanticDistance()
SGSKB_GetWordPos()
SGSKB_GetWordPosAll()
SGSKB_GetWordSuperClasses()
SGSKB_RatePosNeg()
SGSKB_RatePosNegFile()
etc ...
+ many other traversal functions and NLP functions that are built upon the base API

Summary

Text search, text analytics and information retrieval in general have come a long way. There is still much to do to enable truly intelligent and accurate processing of text especially when the data volumes are very large. Progress in this area will continue to require advanced semantic analysis approaches to achieve even moderate increases in accuracy. SGSKB™ has been designed to provide advanced and highly performant techniques to aid both researchers and solution architects.

Bibliography

Alpha, Shamim; Dixon, Paul; Liao, Ciya; Yang, Changwen (2001). Oracle at TREC 10: Filtering and Question Answering. *Proceedings of The 10th Conference on Text Retrieval (2001)*.

Beckwith, Richard (1993). *Design and Implementation of the WordNet Lexical Database and Searching Software.* (From the WordNet home page).

Brill, Eric; Lin, Jimmy; Banko, Michael; Dumais, Susan; Ng, Andrew (2001). Data-Intensive Question Answering. *Proceedings of the 10th Conference on Text REtrieval (2001)*.

Burger, John D (2006). MITRE's Qanda at TREC-15. *Proceedings of the Fifteenth Text REtrieval Conference (TREC-15, 2006)*

Burke, Robin D.; Hammond, Kristian J.; Kulyukin, Vladimir, A.; Lytinen, Steven L.; Tomuro, Noriko; Schoenberg, Scott (1997). Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. *AI* (1997).

Cardie, Claire; Wagstaff, Kiri (1999). Noun Phrase Coreference as Clustering. *Proceedings of the Joint SIG DAT Conference on Empirical Methods in Natural- Language Processing and Very Large Corpora,* 82-89, Association for Computational Linguistics, (1999).

Dagan, Ido; Lee, Lillian; Pereira, Fernando (1997). Similarity-Based Methods For Word-Sense Disambiguation. *Proceedings of The 35th Annual Meeting of The Association of Computational Linguistics and 8th Conference of The European Chapter of The Association For Computational Linguistics (1997).*

Hovy, E.H., L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran. (2001). Toward Semantics-Based Answer Pinpointing. *Proceedings of the DARPA Human Language Technology Conference (HLT)*.

Ittycheriah, Abraham; Franz, Martin; Roukos, Salim (2001). IBM's Statistical Question Answering System. *Proceedings of the 10th Conference on Text Retrieval (2001).*

Khan, L., D. McLeod, and E.H. Hovy. (2004). Retrieval Effectiveness of an Ontology-Based Model for Information Selection. *Journal for Very Large Data Bases (VLDB). 13*(1), 71–85.

Lytinen, Steven L.; Tomuro, Noriko; Rapede, Tom (2000). The Use of WordNet Sense Tagging in FAQFinder. *AAAI-2000 Workshop on AI and Web Search (2000).*

Lytinen, Steven L.; Tomuro, Noriko (2002). The Use of Question Types to Match Questions in FAQFinder. *AAAI-2002 Spring Symposium on Mining Answers From Text (2002).*

Mihalcea, Rada; Moldovan, Dan I., (1998). Word Sense Disambiguation based on Semantic Density. *COLING/ACL (1998).*

Miller, George A. (1993). Nouns in WordNet: A Lexical Inheritance System. From the WordNet home page. Update of original paper (1990). *International Journal of Lexicography, 3(4), 245-264.*

Miller, George A.; Beckwith, Richard; Fellbaum, Christiane; Gross, Derek; and Miller, Katherine A. (1993). *Introduction to WordNet: An On-line Lexical Database.* From the WordNet home page. Update or original paper (1990) *International Journal of Lexicography, 3*(4), 235-244.

Mlynarczyk, Stanley J.; Lytinen Steven L. (2005). FaqFinder Question Answering Improvements Using Question/Answer Matching. *Proceedings of the 2nd Language and Technology Conference (2005)*.

Mlynarczyk, Stanley J., (2009). Investigations of mechanisms to improve question answering. Thesis, DePaul University *(2009)*.

Moldovan, Dan I; Mihalcea, Rada (1999). *Improving the search on the Internet by using WordNet and lexical operators*. Unpublished (1999).

Resnik, Philip (1999). Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence - 1999*, *11*, 95-130

Stetina, Jiri; Kurohashi, Sadao; Nagao, Makoto (1998). General Word Sense Disambiguation Method Based on a Full Sentential Context. *Workshop COLING/ACL (1998).*

Tomuro, Noriko (2002). Question Terminology and Representation for Question Type Classification. *19th International Conference on Computational Linguistics (COLING '02).*

Toutanova, Kristina; Manning, Christopher D. (2001). Feature Selection for a Rich HPSG Grammar Using Decision Trees. *CoNLL-2001.*

TREC-10 (2001). NIST Special Publication 500-250: The Tenth Text REtrieval Conference (TREC 2001) – http://trec.nist.gov/pubs.html.