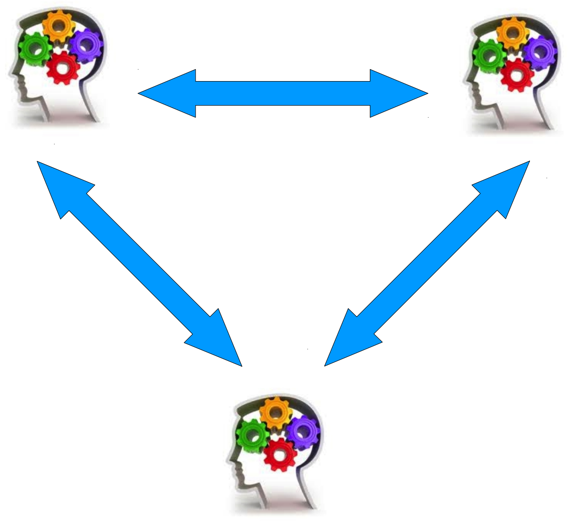
SemantiGrid™

A Modern Text Analytics Platform

Chicago Technologies Incorporated

Abstract

Text analytics is increasingly becoming a focus of the business intelligence community. Whereas BI 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. Functions such as document storage, search and sentiment analysis have become important tools to provide the differentiating edge that may spell success or failure within the marketplace. Other businesses see the benefit of applying new technologies such as BigData platforms to bring additional operational capability that was not possible using non-parallel platforms. SemantiGrid™ is a response to address these needs using the latest capabilities garnered from the research community while also addressing the short-comings of existing BigData platforms and tools that have surfaced to meet these needs. Lastly, it is designed to scale to data levels that far exceed current BigData platforms and to provide the processing ability to allow efficient/performant analysis.



2014 Copyright - Chicago Technologies Incorporated

Why A New Platform

The landscape of data is changing. Over the last 5 years, there has been a general admission that database technology is not able to meet the requirements for processing the variety of data that our world is generating. Not longer are the requirements solely tied to traditional structured data that is commonly found in our transactional databases and enterprise data warehouses. There is a need to deal with text, documents, log files, images, speech as but a few examples. Analysis of these data types require new tools and approaches that go well beyond SQL

Text document analytics is a BigData problem in the sense of storage requirements. Text analysis, while sometimes addressed with database-centric applications, is not ideally suited for SQL platforms. Hadoop too is sometimes considered to possess text analytic capability via Hive and Spark. These capabilities are however very elementary. The Hadoop ecosystem is largely implemented using Java which is a wonderful high level language but too inefficient for the complex processing required for near-real-time text analytics/NLP.

There is also the need to store documents in order to analyze them. It would be very inefficient to store these in an SQL database. Storing documents within the Hadoop File System (HDFS) which has been engineered to efficiently process very large files also has its shortcomings. In a text document store, there are often vast numbers of small documents. These small files of text require many supporting files during processing. While from a total storage requirement, this is BigData, the mechanics of the data are not always conducive to a Hadoop/HDFS implementation. It is for this reason that ElasticSearch has elected to use their own file system implementation as an option and why document storage solutions such as MongoDB have gained traction.

There are a variety of approaches to document document/text analytics within the industry. One traditional approach is to utilize SQL for the analytics functions. This often is tried even on Hadoop within a Hive paradigm. This can work for data volumes that are modest and can scale with MPP databases (such as Netezza, Teradata, Db2, etc...) to a certain extent. One challenge in this instance is the cost. Many of these platforms are expensive compared to a Hadoop/Elastic Search/Solr/MongoDB (or SemantiGrid™) approach because their application space and cost model is business analytics in most cases. There is also a limitation on the level of scaling that can be achieved and the resultant performance even with MPP SQL. The reason for lack of scaling is that as the level of accuracy and sophistication of text analysis increase, SQL and the underlying database architectures become a bottleneck.

SemantiGrid™ has been designed specifically to address the issues around scale/function/performance of both modest and very large (exabyte level and beyond) document management and text analytics. Unlike some of the tools that promise to provide analytics, SemantiGrid™ is a complete production quality solution.

A Complementary Platform To Hadoop and Databases

While SemantiGrid™ has been designed to provide functionality that does is not currently provided in large systems such as Hadoop and EDWs, it can be a "complementary" technology to a Hadoop, Teradata, DB2, Oracle or Netezza platform for example. SemantiGrid™ may be integrated with these systems to provide an advanced level of text analytic capability that would not be possible otherwise. In a hybrid implementation, SemantiGrid™ may be integrated via UDFs and C/C++ or Java JNI that allow the creation of an integrated deep analytics platform. This is addressed below and in the Teradata Integration white paper.

The Need For Automation

One of the considerations in a text analysis platform is how easy it is to use. SemantiGrid™ has been designed to provide automation of many tasks. One area of operational automation is during document ingest. When a document is inserted, SemantiGrid™ will perform many of the preprocessing steps that will prepare it for subsequent analytics. Below are (some) of the operations that will be performed automatically on ingest:

- Word frequency analysis
- Word index generation (node local)
- Cross-document correlations
- Master index builds (across the entire SemantiGrid™ domain)
- Phrase analysis
- Text grammar pre-parse
- Sentiment analysis
- Document classification
- … Plus many others

The goal is to fully prepare a document for analysis and search. In contrast to a framework such as MongoDB (which is little more than a document catalog), SemantiGrid™ provides significant (deep) analytic capability.

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 slugger had his head on straight this afternoon, the contest would have ended much differently”. For most Americans, the above sentence is perfectly clear. Computationally, this same sentence is undecipherable without advanced analytics. We need to understand that slugger refers to a baseball player. This is no small task given that the word can also mean “one who hits another person”. The phrase “head on straight” is equally problematic and must be interpreted to mean “in good form” or “competent”. “Contest” in this case must be equated to baseball game, however, how can we do this programmatically unless we somehow tie this in with the concept of slugger being a baseball player and not someone who hits another person. Lastly, the slugger must be correlated with the actual player name that surely appeared in a prior sentence.

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). For this reason, SemantiGrid™ has been fashioned to utilize semantic methods to provide a much deeper analysis of text and yield increased accuracy in text matching.

Semantic Foundation

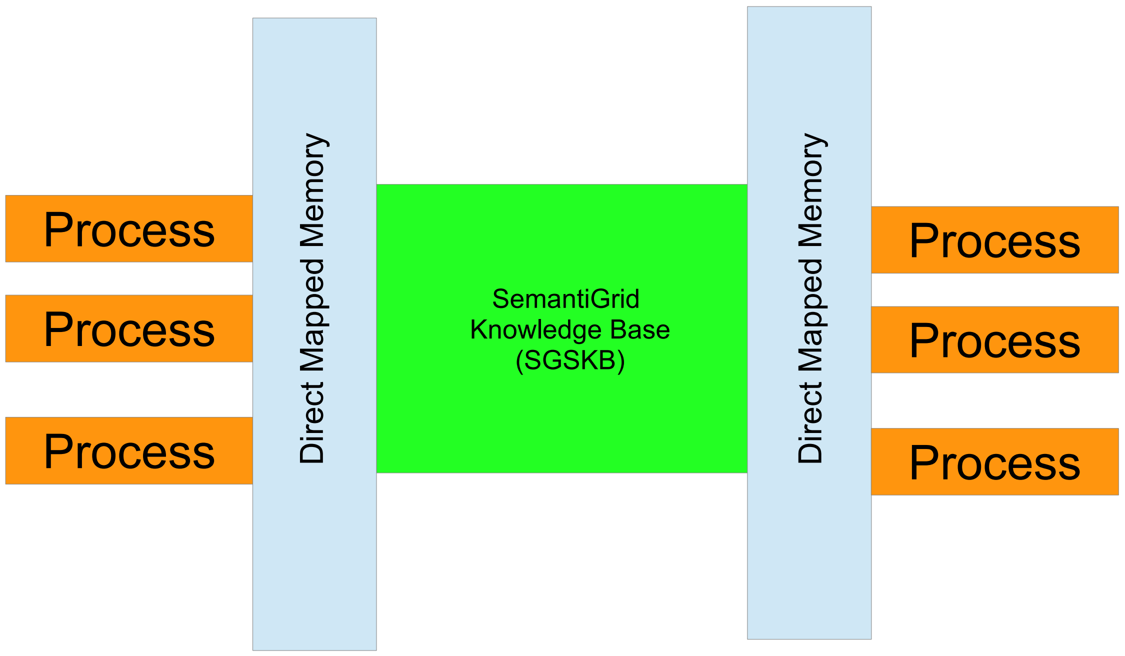
SemantiGrid™ was designed to generate enhanced accuracy. This includes utilizing the simple methods (text match and word frequency for example) but augmented with semantic methods that are engineered to significantly increase the accuracy of text match. These semantic methods include: synonymy, word sense, semantic word similarity/distance, Subject/Verb/Object and phrase parsing/analysis (resolution of “head on straight” as an example).

While semantic methods are able to provide the enhanced accuracy that is lacking in simple word match approaches, the challenge becomes efficient processing on a computational platform. ***A vast amount of computational resources is often required to implement semantic processing***. The resources must be utilized in a parallel fashion where a given problem is partitioned into units of parallelism that are convenient to distribute across a computational cluster. As important as parallelism, there is a need to make certain high-frequency operations efficient – such as word ontology lookup. It is for this reason that an in-memory Knowledge Base is fundamental for high frequency lookup. SemantiGrid™ has been designed to be parallel in its operation but also highly efficient in its execution of logic that can be a bottleneck in performance (the Knowledge Base).

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 compete Webster Dictionary, synonym relationships and word relationship (semantic similarity, senses, ...). The repository is a modified tree structure augmented with additional relationships to form the N-Dimensional surface features. The memory resident repository allows multi-process/multi-user direct access. This approach provides extremely fast access, search and relationship analysis.

The performance of the SGSKB™ is not simply due to its memory residency. It is the relation-based structure of the SGSKB™ ontology and its unique traversal functions that yield performance that is on par with hardware-based (silicon) implementations.



The above graphic 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 the SemantiGrid™ cluster (or Hadoop cluster or EDW) 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. It 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, ...)

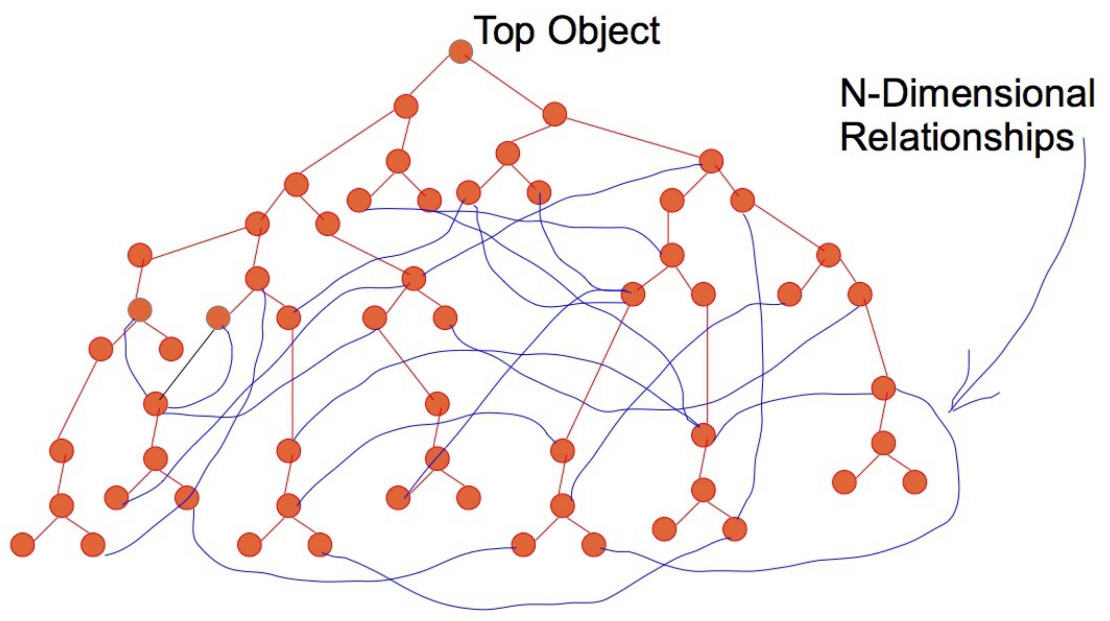
Why have a knowledge base when there are approaches using a database for operations such as synonym lookups or computation of word sense or word distance? While it is true that one can take a database/SQL approach for this task, a major consideration will be scale. If the document store to be scanned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

2014 Copyright - Chicago Technologies Incorporated

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. Some of the details of the SGSKB™ implementation include:

- Multi-tenancy (shared copy of the knowledge store)
- Ultra fast accesses (usually sub microsecond lookups)
- Tree structure augmented with N-dimensional relationships
- Currently there are about 188,000 nodes in the SGSKB™
- There are hundreds of thousands relationships within the knowledge base
- Support for adding additional data domains such as medical or scientific.
- Support for additional languages (other than English).
- Support for phrases (comes with over 50,000 phrases)
- Support for ISA relationships
- Support for sentiment
- Support for classification
- Part of speech support (full dictionary)
- Fuzzy match (ultra fast because of the SGSKB™)
- Antonyms
- New SGSKB™ relationships can be added either via user interface or via batch load
- User/Administrative tool
- Self synchronizing across a cluster

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 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:

2014 Copyright - Chicago Technologies Incorporated

- 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. It clearly demonstrates the adherence to a policy of "no system calls" and "no context switches" to be able support such high rates of 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 gives 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.

Extensibility

SemantiGrid™ has been engineered to support a variety of document storage needs. Whether it is a city, county or state court case system, email repository, medical document storehouse, company documents or library, SemantiGrid™ supports the hierarchical relationships that may be required along with access controls and ease of access for any document storage need. Its growth is linear, predictable, secure, performant, reliable and cost effective. SemantiGrid™ is capable of storing any document or data object including pictures, voice and backup/archives from databases. Large objects are split across nodes, small objects are stored efficiently in their entirety on a given node (plus other nodes for redundancy). The ability to manage both large objects and small objects efficiently is a key consideration for using SemantiGrid™.

Business Production Focus

The challenge of using existing options for text analytics (such as Hadoop) is that they were never designed for a business production environment where high security (HIPAA or PCI compliance for example), traceability, 0-data risk, updatability of data, ease of use, ease of implementation were necessary ingredients. There are concerted efforts to enhance Hadoop and there is much activity surrounding tools for the Hadoop environment.

SemantiGrid™ on the other hand has been designed with a business production environment as a requirement. Its support of data encryption, secure authentication, access logging, user/group/application access to documents, synchronized multi-site capability providing 0-data risk, self healing on component failure, rich administrative capability that integrates with existing system/network managers, ease of growth (just add a data node and it will be seamlessly integrated into the fabric), automatic data balancing across nodes and highly efficient inter-node data communication (based on UDP broadcast) as but a few of the considerations that are core to SemantiGrid™'s design.

Thought has been given to document/object organization. For applications such as medical record or legal record storage, SemantiGrid™ has been designed with the concept of a master-child hierarchy so that documents can be organized within a case or patient paradigm (all information whether master or child is uniformly an object). This also implies that a single encryption key can be used to manage a "dossier" of documents but also provides the ability to have a separate encryption key for any individual document within the hierarchy.

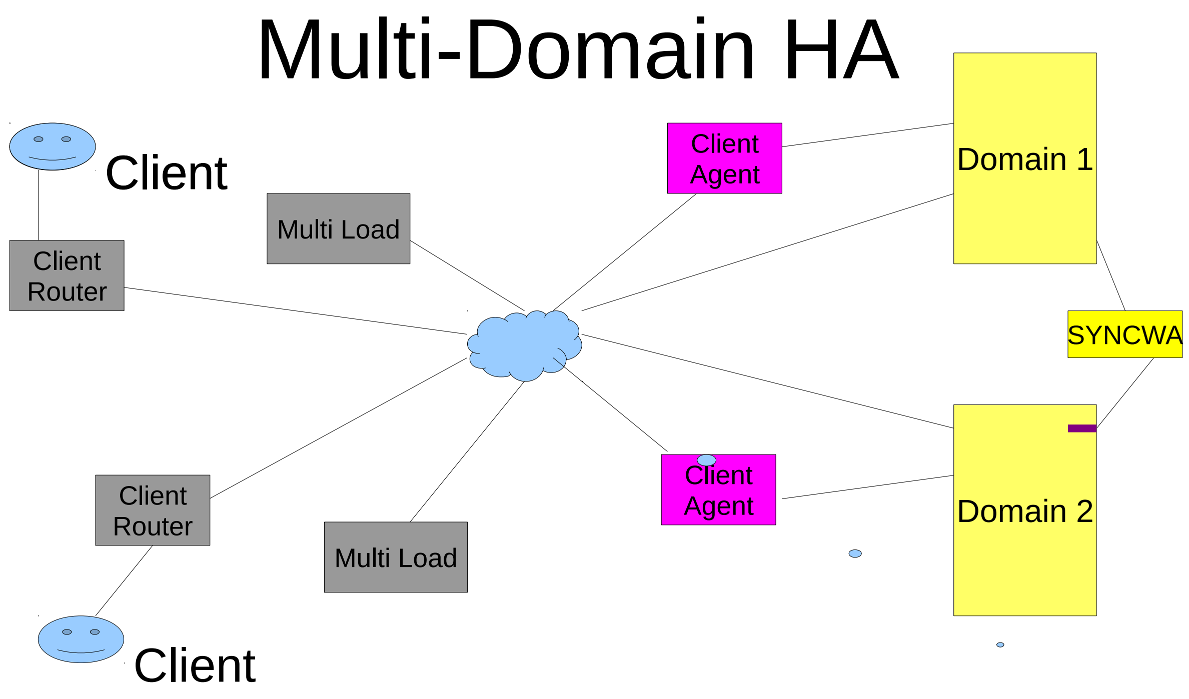
There is also the ability to protect documents at the user, group, case/project or object level from unauthorized access. Access logging is available for all of these or can be relaxed for less stringent security needs. The security model supports very granular access as well as broad access for analytics across users/projects/objects if that is the desired functionality.

Data Protection

SemantiGrid™ has been designed with data protection as a key feature. Data (objects) are replicated via a system default (usually 3 nodes), however, there is the ability to specify replication at a document level or a individual user/group level. This permits for example the ability for a given document to be resident on all nodes if desired or to have no replication for a given user community if that is desired. Loss of a node is transparent to the function of the system and the system has the ability to heal from such a failure (data that resided on the failed node will be repopulated on surviving nodes). Each node is individually able to function even if all other nodes in the system are down and will respond to queries. Given the manner in which documents/objects are distributed, the system could suffer a catastrophe where a majority of the nodes were lost, yet the data on the surviving nodes would still be viable.

Support for multi-site (multiple domains) is supported. Use of this feature will ensure that the loss of a data center will not result in the loss of data. This synchronization is managed in multiple ways. First, during submission of a document to SemantiGrid™, the submitter may elect to have the submission span another domain. This will ensure that the object/document will be dual loaded to another domain ***(dual load is more difficult to implement, however, it is the most efficient and reliable mechanism of synchronization)***. The dual load can occur within client object operations where low volume but also low latency are required.

SemantiGrid™ also supports the multi-domain concept via its "multiload" service that can be used as front end to bulk data loads into the system. Multiload will ensure that all newly instantiated objects are sent to as many domains/clusters as are defined even if those domains/clusters are not currently available (the updates are held for the inaccessible system until communication is restored).



Availability

In concert with the ability to load multiple domains, there is a query router that can be used to distribute queries across multiple domains. A user connects to a query router (there can be multiple query routers defined for performance and availability) and the router can route the user to a given domain depending on the routing rules set for that user. For users who have multi-site/multi-domain data, the query router can transparently route a user request to an active domain when another domain is down.

A query router's function is controlled with a set of rules. These rules can be at a domain level or node level for the destination and are per-user at the client end. For example:

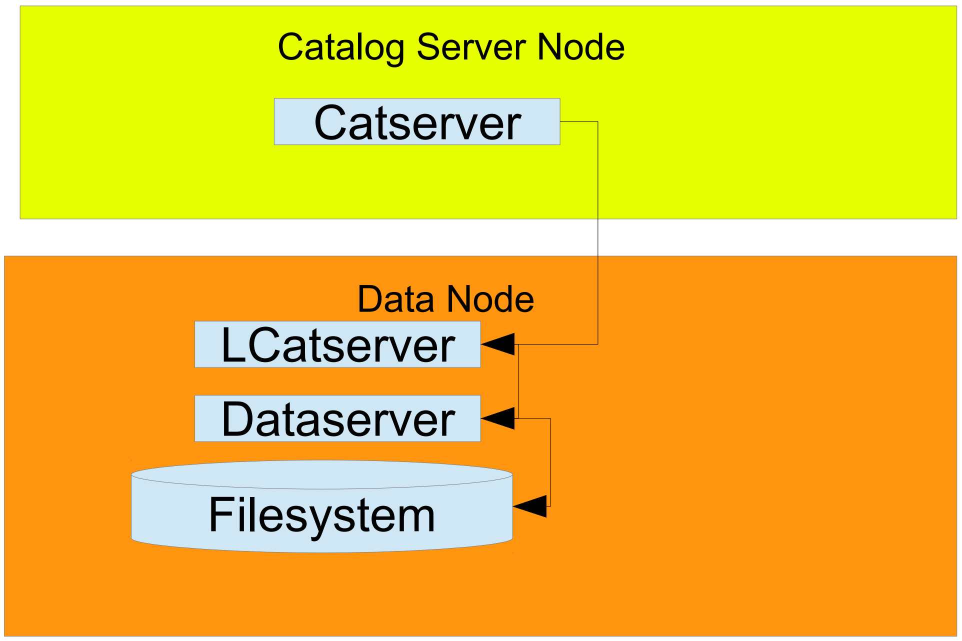
bjohnson	newyork.*,chicago.*:PREFERRED
msmith	newyork.node001,node002.chicago.*:PREFERRED
CRM	newyork.*,chicago.*:WEIGHTEDROUNDROBIN:80,20

In the above, bjohnson can be routed to any node in the “newyork” cluster as a preferred location. In the event of a problem with access to “newyork”, bjohnson will be routed to the chicago cluster.

Msmith on the other hand will normally be routed to one of two nodes on the newyork cluster but will be routed to any available node on the chicago cluster. Please note that the client software connection layer will poll a destination for available nodes if the rules do not specify a specific target node.

In the third example, the CRM application connections will be distributed across the two domains using a weighted round-robin approach where 80% of the connections will be sent to newyork and 20% to chicago.

Complementing the above cross-domain resiliency is the design of the catalog system (Catservers). Catservers store information about users/objects. There is usually more than one catserver in a configuration and they are synchronized. Failure of all Catservers will not prevent user access to data. Access may be slightly degraded from a performance standpoint because applications will be routed to data nodes where the catalog information is maintained at a node level within local catservers. There are therefor three levels of protection of catalog information:



Both Cat and LCatServers maintain their meta data in SQL databases. Should all of the Catalog Servers fail, SemantiGrid^{TM} will automatically and transparently contact the LCatServers that reside on the data nodes for assistance in obtaining information about an object. Should the LCatservers be inoperable, SemantiGrid will automatically and transparently communicate directly with the DataServers to provide information about objects. The most efficient mechanism is to have the CatServers provide this service, however, the system will provide progressive recovery in the event of serious system issues. The recovery does not stop at the software level however. SemantiGrid™ can recover all of its data on a node even when all Cat and LCatServer information is lost. This is accomplished by a rebuild of the LCatServer's data repository from the file system. All object information is maintained on disk and thus the Catalog servers' information serves to provide enhanced performance without being a weak link from a reliability standpoint.

To further illustrate this point, in the unlikely event that all of the nodes are destroyed in a catastrophe save for one node, that one remaining node's data will (not) be lost --- of course, a secondary SemantiGrid™ domain offers data-center level of protection that would guarantee 0-data risk from a site catastrophe.

CatalogServer Synchronization (Patented)

Synchronization of CatServers and LCatServers is accomplished with a ***patented*** low latency coherency protocol. This has the benefit of virtually instantaneous sync state determination. Additionally, the same protocol will allow for very fast resynchronization of CatServers or LCatServers. Without this capability, growth of the system would be jeopardized as the object store becomes very large. The CatalogServer's unique synchronization and the system's ability to function with a degraded CatServer configuration or complete lack of CatServers, support enterprise class system and application availability.

Automated Document Classification

As documents (objects) are added to SemantiGrid™, they are automatically managed by the system (the NLP service). A document may be a single file, however, SemantiGrid™ (nlpserver) will generate a number of supporting structures. This includes but is not limited to the following:

- Original document
- Scrubbed text
- Meta data for CatalogServer regeneration (creation information)
- Word Frequency files
- Multi-word analysis
- Annotations as a result of semantic parsing
- Sentence and word level senses
- Document classification
- Multi-version support
- Subject/Verb/Object analysis
- Various indexes to enhance semantic search

All of the above are managed by the system without the need for intervention by any user application.

Some of this information is also then placed within the local SQL database for enhanced access not only within the local node but also for access across the system (such as for stats). Note that the creation of the ancillary information occurs as a separate step as a post-load operation. This ensures a very high rate of load capability while still providing the additional access/analytic support over the data.

Post-load operations are a process that is ongoing. The data-store is periodically scanned for consistency to ensure that a given object has been processed appropriately and that the necessary support data is present.

Health Checks

Health and sanity checks are periodically run as follows:

1) On inbound directories/persistent queues to detect and correct anomalies. All inbound jobs and object creations are logged within disk fifos to ensure consistency even with system failures during these operations.
2) All job process steps are logged on disk to ensure they are restartable.
3) All objects (structures) are periodically scanned for consistency and any anomalies reported and corrected.
4) All CatalogServer and LCatServer data is checked for consistency. This is done using a patented fast-check approach that ensures consistency and fast recovery even over very large numbers of objects.
5) All CatalogServer and LCatServer data is checked for consistency against the actual object repositories to ensure consistency.
6) File system checks to protect against disk irregularities.
7) Node checks to detect node failures.
8) All SemantiGrid^{TM} processes are monitored. This includes not only whether a process is active but also whether it is responding to a health check request. This is done at a node level and any anomalies are reported to the CatServers and Monitoring Control server.
9) All jobs are checked/tracked for completion and status.

Programmatic and User Data Access

In most cases, users will access data via a client that understands data placement within the cluster. There is however an important aspect of data visibility that pertains to both client software and end users of data; both would like to know and access their data easily. For this reason, all SemantiGrid^{TM} data is stored as simple files within the LINUX file system. This permits the use of standard file system operating system calls (open, read, write, fopen, …) and makes programming a more pleasant experience.

Because the data is housed within the LINUX file system, this opens up the possibility of allowing users to “see” their files as normal file system-organized information (directory/files). This is made possible by setting up a user's view of their data via symbolic links that exist on a per-node basis.

2014 Copyright - Chicago Technologies Incorporated

There is also the ability to have a file system server where all nodes' data is consolidated and viewable by standard file-manager/shells.

A SemantiGrid™ data organization is much more extensive than just the data files and includes many ancillary objects – such as word frequencies and indexes. These are available to SemantiGrid™ functions but are normally of no interest to users or even application code as application code will normally access data using predefined functions. These (could) be accessible if desired however.

Network Performance

In addition to the analytics software, there are two additional aspects of performance that play major parts in a text analytics platform; the platform itself and the protocols of the distributed system. It is for this reason that SemantiGrid™ supports multiple networks and network interfaces for different operations within the cluster. The data nodes for example use a dedicated network for transfers of large amounts of data. Task servers utilize this same method for communicating parallel execution requests to target nodes/agents. There is also additional support for network interfaces between the catalog servers and clients.

Why so many interfaces? *(NOTE: SemantiGrid™ will function with a single interface if that is desired)* It is not just raw throughput that is the consideration. The second aspect of SemantiGrid™ performance is the protocols used for inter-node communication and how this takes advantage of the multiple interfaces. Unlike TCPIP, where collisions and contention can be hidden from the user, UDP (Universal Datagram Protocol) is a connectionless protocol that provides the ability to broadcast data to multiple servers (in parallel) on a network at the expense of a possibility that the broadcast may not reach its intended target (SemantiGrid™'s protocol assumes this and has built in re-transmit to ensure reliable operation).

The reason UDP protocol is critical in a multi-node environment is because of the performance benefit in being able to send a message (once) to many target nodes. For example, SemantiGrid™ transmits data to multiple data nodes using a targeted transmission where the targeted nodes all receive the transmission in parallel, thereby eliminating the need to transmit the data individually to each node. The parallel execution protocol functions similarly (SemantiGrid™ supports both broadcast and multicast parallel execution of logic across the cluster). Catalog servers also communicate in this way with all of the data nodes to maintain status. Having multiple network interfaces in combination with use of UDP has a significant performance benefit and facilitates much larger cluster growth; something that has been problematic with some of the larger Hadoop implementations.

To summarize – SemantiGrid™ has been designed to scale far beyond Hadoop-based implementations. From network communications, to the in-memory knowledge base and semantic functions, plus careful design of software components, maximizes the performance that is possible from a parallel platform. Scaling is ***not*** just about the ability to store data. Having the technology to effectively analyze extremely high volumes of data is equally important. Advanced semantic text analytics technology residing on a performant, reliable, operationally low cost and secure platform is where SemantiGrid™ is at its best.

Security and Access Logging

An important aspect of many kinds of documents/data is the security surrounding access/authentication to the data. This includes not access protection of the data but also the maintenance of access logs that provide detail around who has accessed a particular portion of data and when. SemantiGrid™ provides the following:

- Encryption at a user level – Single default encryption key
- Encryption at a document level – A key that is dedicated to a specific document
- Encryption at a group level – A default shared key that applies across a user group
- Encryption at a system level – A default shared key for all documents within the cluster
- No encryption – Where the data is not sensitive and can be shared by all
- Access logging system level - All access is logged (user, document, date/time)
- Access logging user level – All access to a specific user's data (user, document, date/time)
- Access logging group level – All access at a group level (user, document, date/time)
- No access logging

Access/authentication is the first level of security and SemantiGrid™ provides a key-based authentication mechanism that ensures a user is who they say they are. This is important as even though there may be other security measures enabled (encryption/logging), there is still the need to make sure that a malicious use does not “destroy” data. Strong authentication is the first line of defense.

Encryption is critical to ensuring adherence to industry security compliance standards such as HIPAA, PCI and government standards. SemantiGrid™ provides a very flexible encryption approach from system defaults to document specific access restrictions. Encryption can be selected to at a system, group or individual user level and is allowed to vary user-to-user and document-to-document. This provides both excellent compliance where called for, but also allows data to be easily accessible in less restrictive data domains.

Once the authentication and access to data are in place, many high security environments require that authorized users' access to individual documents be tracked. The medical, government and financial domains are areas where compliance with data access are especially important and in many cases mandated. Access logging is a mechanism that completes the circle of security control and SemantiGrid™ provides this and the other critical security features as part of the product core.

Messaging Protocol

Cluster agents communicate using both UDP and TCP. In both cases, delivery of messages is guaranteed or will immediately be known to have failed as this type of communication is point-to-point. This supports the messaging parallelism required along with very high-frequency of transmission (messages/packets are tuned for high speed deliveryand vary in size depending on message type).

Another level of messaging is necessary however within the cluster. This messaging requires

2014 Copyright - Chicago Technologies Incorporated

guaranteed delivery even if the target or network is not currently available. Event/Alert messages are an example of this as it is critical that these not be lost.

While a message queue can mitigate some of this concern, there is also the need to administer and protect the queue. SemantiGrid™ employs a messaging protocol that ensures (eventual) delivery of all such messages. This requires no queue mechanism yet guarantees delivery along with high frequency of message transmission (thousands per second).

Hadoop Coexistence

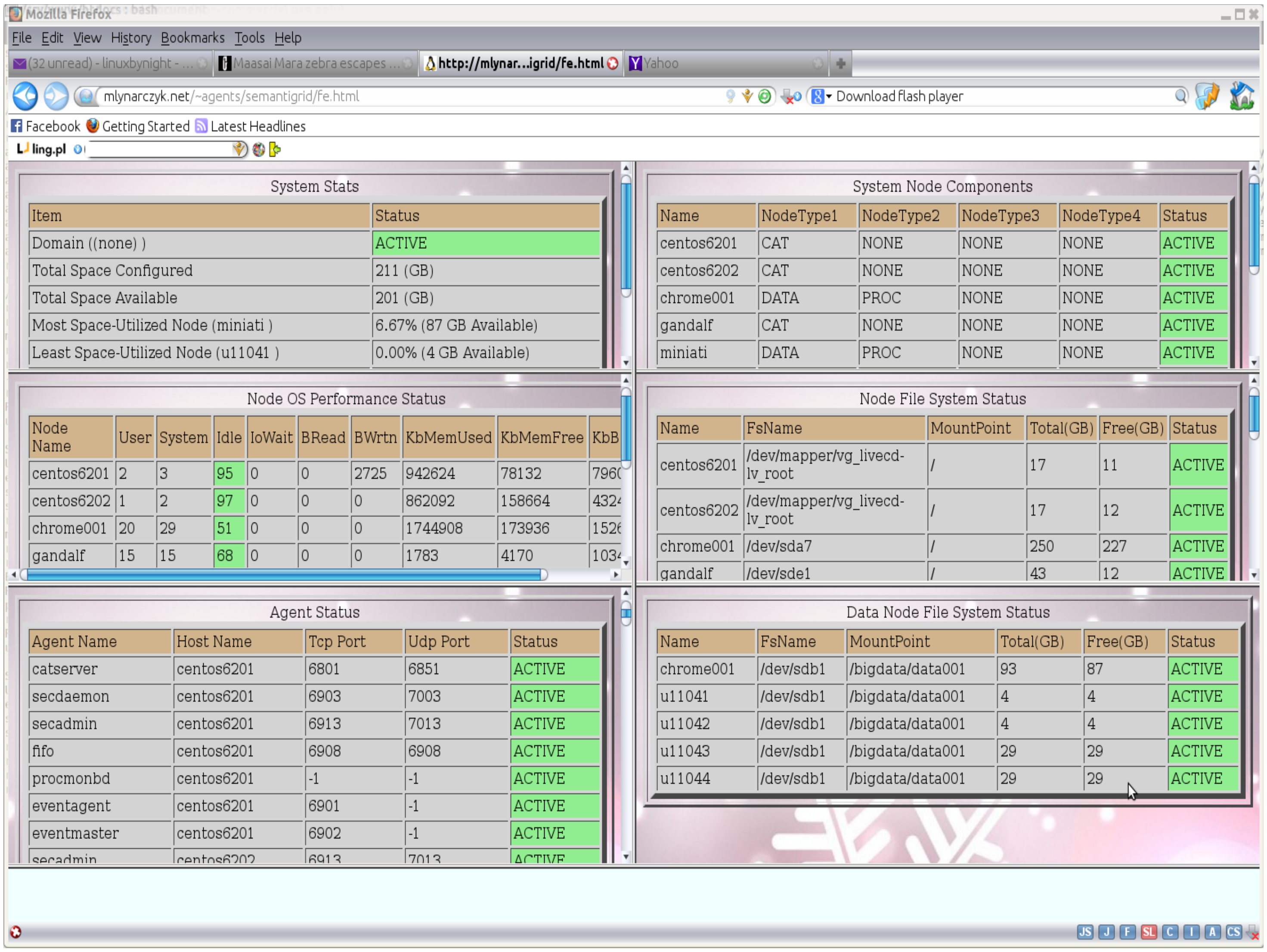
SemantiGrid™ does not require any Hadoop functions or functionality. It can however be installed on an existing Hadoop cluster in a coexistence configuration. Given the large IO requirements of Hadoop, it is recommended that dedicated storage be used for SemantiGrid™ data if a coexistence model is desired. A secondary dedicated network interface for SemantiGrid™ is recommended but not required.

Integration with Hadoop may be accomplished using the available library of Java JNI functions that provide for a highly performant implementation. There are also client utilities that allow for a non-programmatic integration that provide for rapid implementation.

Administrative Functions

Control of the SemantiGrid™ cluster is automated to a large extent in that errors are handled by software and recovery actions are automated. An administrative interface provides information on system state, performance, operating system stats, space utilization, object storage information, tasks, users + cluster startup/shutdown and software upgrade, etc.... Below is a snapshot of the system status menu for an active SemantiGrid™ cluster.

2014 Copyright - Chicago Technologies Incorporated


Mozilla Firefox
File Edit View History Bookmarks Tools Help
(32 unread) - linuxbynight - ...
Maasai Mara zebra escapes ...
http://mlynar...igrid/fe.html
Yahoo
mlynarczyk.net/~agents/semantigrid/fe.html
Download flash player
Facebook
Getting Started
Latest Headlines
ling.pl

System Stats

Item	Status
Domain ((none))	ACTIVE
Total Space Configured	211 (GB)
Total Space Available	201 (GB)
Most Space-Utilized Node (miniati)	6.67% (87 GB Available)
Least Space-Utilized Node (u11041)	0.00% (4 GB Available)

System Node Components

Name	NodeType1	NodeType2	NodeType3	NodeType4	Status
centos6201	CAT	NONE	NONE	NONE	ACTIVE
centos6202	CAT	NONE	NONE	NONE	ACTIVE
chrome001	DATA	PROC	NONE	NONE	ACTIVE
gandalf	CAT	NONE	NONE	NONE	ACTIVE
miniati	DATA	PROC	NONE	NONE	ACTIVE

Node OS Performance Status

Node Name	User	System	Idle	IoWait	BRead	BWrtn	KbMemUsed	KbMemFree	KbB
centos6201	2	3	95	0	0	2725	942624	78132	796
centos6202	1	2	97	0	0	0	862092	158664	432
chrome001	20	29	51	0	0	0	1744908	173936	152
gandalf	15	15	68	0	0	0	1783	4170	103

Node File System Status

Name	FsName	MountPoint	Total(GB)	Free(GB)	Status
centos6201	/dev/mapper/vg_livecd-lv_root	/	17	11	ACTIVE
centos6202	/dev/mapper/vg_livecd-lv_root	/	17	12	ACTIVE
chrome001	/dev/sda7	/	250	227	ACTIVE
gandalf	/dev/sde1	/	43	12	ACTIVE

Agent Status

Agent Name	Host Name	Tcp Port	Udp Port	Status
catserver	centos6201	6801	6851	ACTIVE
secdaemon	centos6201	6903	7003	ACTIVE
secadmin	centos6201	6913	7013	ACTIVE
fifo	centos6201	6908	6908	ACTIVE
procmonbd	centos6201	-1	-1	ACTIVE
eventagent	centos6201	6901	-1	ACTIVE
eventmaster	centos6201	6902	-1	ACTIVE
secadmin	centos6202	6913	7013	ACTIVE

Data Node File System Status

Name	FsName	MountPoint	Total(GB)	Free(GB)	Status
chrome001	/dev/sdb1	/bigdata/data001	93	87	ACTIVE
u11041	/dev/sdb1	/bigdata/data001	4	4	ACTIVE
u11042	/dev/sdb1	/bigdata/data001	4	4	ACTIVE
u11043	/dev/sdb1	/bigdata/data001	29	29	ACTIVE
u11044	/dev/sdb1	/bigdata/data001	29	29	ACTIVE

A full suite of control functions are available to the administrator including for example:

- Cluster Start/Stop
- Node Start/Stop
- Agent Start/Stop
- Job Start/Stop/Suspend/Restart
- Cluster or Node State Change - ACTIVE/STANDBY/OOS (Out of Service)
- Enable/Disable User
- Enable/Disable Group
- Add a Node
- Delete a Node

2014 Copyright - Chicago Technologies Incorporated

- Restore a Node
- Run System/Node/Data checks
- Cluster/Node/Agent/Job Status
- View Event Logs
- ...

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. Just as important however, is that this functionality be performant. SemantiGrid™ has been designed to provide these advanced techniques in a platform that can scale far beyond current grid technologies while offering the types of features required for enterprise class business applications.

2014 Copyright - Chicago Technologies Incorporated

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).*

2014 Copyright - Chicago Technologies Incorporated

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 Unversity *(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.

2014 Copyright - Chicago Technologies Incorporated