The Semantic Vectors Package: New Algorithms and Public Tools for Distributional Semantics

Fourth IEEE International Conference on Semantic Computing (IEEE ICSC2010),
Carnegie Mellon University, Pittsburgh, Pennsylvania, September 22-24, 2010.

Dominic Widdows
Google, Inc.
widdows@google.com

Trevor Cohen
University of Texas
trevor.cohen@uth.tmc.edu

Abstract—**Distributional semantics is the branch of natural language processing that attempts to model the meanings of words, phrases and documents from the distribution and usage of words in a corpus of text. In the past three years, research in this area has been accelerated by the availability of the Semantic Vectors package, a stable, fast, scalable, and free software package for creating and exploring concepts in distributional models.**

This paper introduces the broad field of distributional semantics, the role of vector models within this field, and describes some of the results that have been made possible by the Semantic Vectors package. These applications of Semantic Vectors have so far included contributions to medical informatics and knowledge discovery, analysis of scientific articles, and even Biblical scholarship. Of particular interest is the recent emergence of models that take word order and other ordered structures into account, using permutation of coordinates to model directional relationships and semantic predicates.

I. Distributional Semantics and Vector Models

Distributional semantics is an empirical field of research and development that attempts to discover and model the meanings of words by analyzing and comparing their distributions in large text corpora. This approach to meaning can be traced at least to the philosopher Wittgenstein:

> For a large class of cases — though not for all — in which we employ the word 'meaning' it can be defined thus: the meaning of a word is its use in the language. [1, §43]

and the linguist Firth:

> You shall know a word by the company it keeps.[2]

In a sense, this principle has been implemented for centuries in traditional concordances (see [3, Ch1]), and its application to information in electronic form began during the first few decades of research in information retrieval [4], [5]. As with some of the fundamental models for search engines, some of these approaches are consciously probabilistic (representing word distributions as probability distributions), and others are consciously geometric (representing word distributions as vectors in high-dimensional spaces). In recent years, insights from quantum mechanics have suggested that these two mind-frames can be readily derived from common principles [6], but this unification is beyond the scope of this paper, which thus describes developments in geometric models and makes no attempt to describe these models alongside probabilistic alternatives.

The simplest distributional vector model is the *term-document matrix* used by vector model search engines [5],[7, Ch 5]. A term-document matrix is a table that keeps count of how many times each term in a corpus appears in each document. As with any matrix, the rows or columns can be thought of as vectors in some vector space, so in a term-document matrix whose rows represent terms and whose columns represent documents, each row can be thought of as a *term vector* whose dimension (number of coordinates) is equal to the number of documents in the collection.

Not surprisingly, such matrices tend to be very sparse, which invites compression. Sparse representations are used in practice for optimized computation and storage, but computational optimization is not the only motivation. Semantically it is clearly incorrect to assume that each document (or each term) gives a new dimension orthogonal to all others, and in practice, being aware of redundancies here (e.g., cases in which one term can be defined using a combination of other terms) is a core part of human linguistic knowledge. For this reason, matrix factorization and other compression algorithms have been used, beginning with the application of singular value decomposition (SVD) to term-document matrices, in a technique known since the early 1990's as latent semantic indexing or latent semantic analysis [8], [9]. Singular value decomposition produces a reduced set of orthogonal axes for representing term vectors, where the number of these axes (often between 100 and 500) is much smaller than the original number of dimensions (often equal to the total number of documents in the corpus). It has sometimes been demonstrated that words whose vectors are mapped closer together in this decomposition are often closely related and even synonymous. Singular value decomposition is, however, a computationally costly process, normally of time complexity at least $O(qp^2)$ for a matrix of size (q, p) [10, §31]. In addition, the process is memory intensive, largely because the full matrix (before compression) has to be stored in memory.

Random Projection is a cheaper alternative to singular value decomposition which provides many of the same semantic benefits [11]. Instead of spending large computational resources guaranteeing that the basis for the reduced space consists of orthogonal vectors, Random Projection rests on the observation that, in high dimensions, any pair of vectors chosen at random will be *nearly* orthogonal. In practice, an easy way of generating such pseudo-orthogonal vectors is to allocate a vector of zeros, and randomly change a small number of coordinates to $+1$ and the same small number of coordinates to -1. For example, consider the following random vectors

$$A = (0, 0, 1, 0, 1, 0, -1, \ldots, 0, 0)$$

$$B = (0, 1, 0, 0, -1, 0, 1, \ldots, 0, -1)$$

and the scalar product $A \cdot B = \sum A_i B_i$. It is easy to see that most of the individual products $A_i B_i$ are zero, and that the nonzero products are evenly distributed between $+1$ and -1 contributions, leaving an expected total near to zero. An important result that follows from this is the Johnson-Lindenstrass Lemma [12], which guarantees that projection from a high-dimensional vector space onto a smaller subspace is likely to preserve the ordering of scalar products. Thus, Random Projection can be used to reduce the number of dimensions of a high-dimensional space of term vectors, while preserving most of the relationships between those vectors. Note that we should not expect random projection to improve these relationships, for example by mapping synonyms closer together, but this challenge can be addressed using Reflective Random Indexing, as discussed in Section IV-C.

A key observation is that rather than being roughly cubic, Random Projection from a sparsely represented term-document matrix can be made to perform more-or-less linearly [13]. This is a key advantage. A couple of decades ago, we might perhaps have argued that "Moore's law will take careof it." With hindsight, the opposite has happened — the size of linguistic corpora has grown at least as fast as the resources available to any one machine, so algorithms whose complexity is any more than linear in the size of the corpus are becoming gradually less tractable rather than more. This naturally leads to increased interest in algorithms that can take advantage of huge datasets, even if they are less sensitive on small datasets.

Examples of the time taken for matrix compression using Random Projection and SVD on different corpora are shown in Table I. Random Projection always outperforms SVD in this area, and the difference become more pronounced with larger corpora. Note that the numbers are not completely representative of the algorithms in question, because they also include many basic I/O operations. The N/A (not applicable) entry for SVD on 5,000,000 Medline documents is because in this case the workstation ran out of memory (max heap allocation was set to 8GB).

The best way to get an intuitive feel for the power and potential of these distributional semantic methods is to consider some examples, as shown in Table II. The 'RP' column shows results from Random Projection, the 'SVD' column shows results from Singular Value Decomposition. These results are reasonably typical: from manual inspection, we have not yet found queries where these algorithms give results of significantly different quality (though if such results are found, we will endeavor to make them publicly known).

TABLE I
TIME TAKEN FOR THE SEMANTIC VECTORS PACKAGE TO INDEX DIFFERENT DOCUMENT COLLECTIONS USING RANDOM PROJECTION AND SINGULAR VALUE DECOMPOSITION

Corpus	Num Docs	RP Time	SVD Time
King James Bible	1256	2.8s	5.8s
Europarl English	4047	13s	37s
TASA	44,487	21.27s	2m 27s
Medline	1,000,000	1m 17s	6m 19s
Medline	5,000,000	2m 15s	N/A

TABLE II
NEAREST NEIGHBOR TERMS WITH THEIR COSINE SIMILARITIES IN VARIOUS MODELS

King James Bible. Query "abraham".

RP		SVD	
1.00	abraham	1.00	abraham
0.64	sarah	0.81	isaac
0.54	isaac	0.68	sarah
0.47	bethuel	0.62	rebekah
0.44	mamre	0.61	phicol
0.42	jidlaph	0.57	bethuel
0.42	jehovahjireh	0.56	herdmen
0.42	thahash	0.55	nahor
0.42	tebah	0.54	ahuzzath
0.42	pildash	o.54	esek

English Europarl Corpus. Query "wine".

RP		SVD	
1.00	wine	1.00	wine
0.78	wines	0.76	wines
0.77	vineyards	0.63	grapes
0.76	musts	0.54	musts
0.75	vineyard	0.48	winegrowers
0.73	bottling	0.47	litre
0.69	grape	0.46	alcohol
0.68	distillation	0.43	alcoholic
0.67	saccharose	0.42	klerk
0.66	liqueur	0.42	distillation

From results such as these it can be seen that the distributional hypothesis can produce results with clearly semantically significant results, as has been shown in many research experiments (see [7], [14] for many references). However, distributional models have yet to realize their full potential as part of the industrial mainstream. We believe that part of the problem here has been with software performance and reliability: hence much of the research in this paper focuses on this area.

II. THE SEMANTIC VECTORS PACKAGE

The Semantic Vectors package, originally created for the University of Pittsburgh, was released as an open source package in October 2007 [15]. The package can be freely downloaded over the web from http://code.google.com/p/semanticvectors. It is released under a liberal BSD license that permits commercial and non-commercial use and modifications. The goal of the project is to provide a scalable and stable platform for building commercial scale distributional semantics applications, and for researchers and developers to use as a platform for implementing new algorithms. Since the release of the package there have been over 6000 downloads, the development team has slowly grown, and an increasing variety of algorithms and features have been added. To date, these include:

- Sparse representations that scale to corpora containing several million documents.
- In memory index caching, enabling batch mode experiments to scale to thousands of queries.
- Support for bilingual models from parallel corpora.
- Basic clustering and visualization tools.
- Permutation and holographic encoding of word order information.
- Reflective Random Indexing, a new technique for incremental learning.

Vital to the growth of the project, these developments have not come from a single source, but from several contributors from several institutions. The package has supported several experimental analyses that have led to publications and successful dissertation and thesis projects. It has also provided a framework within which research engineers can relatively easily create implementations of recently published research, and so build rapidly upon the state of the art.

This report summarizes the design and development of the Semantic Vectors project, and presents some of the research that has been performed using the package. Most of the features described in this report are released as part of the standard Semantic Vectors package and can readily be demonstrated in action. This article also includes references to other projects that have successfully used the package.

III. Package Design and Development

The Semantic Vectors is implemented entirely in the Java programming language, and depends only upon other libraries written in Java. Here is a summary of some of the key design principles and development activities.

Semantic Indexing. The creation of distributional semantic models using the package proceeds in two phases. First, the user builds a standard term-document index using the Apache Lucene search engine. Apache Lucene is one of the most prominent open source search engines: it is widely available, reliable, well-supported by an active developer community, and enables users to perform many important linguistic operations such as tokenization and stemming in different languages using off-the-shelf components. Second, the user runs the Semantic Vectors indexing algorithm itself, which creates the distributional model by applying Random Projection to the term-document matrix as pioneered by [16]. This process is extremely fast (much faster than building the Lucene index), and has far better scaling properties than Singular Value Decomposition (SVD), used in traditional Latent Semantic Indexing [8]. SVD has recently been added to the package as an option, but it is computationally expensive and is not preferred. In addition to being computationally tractable, Random Projection can be performed iteratively, and it would be comparatively simple to implement both incremental and distributed indexing in this framework. This has not yet been done, largely because the package runs so much faster than Apache Lucene anyway.

The result of semantic indexing is a store of distributional term and document vectors, which can be represented on disk using an optimized Lucene binary format (preferred) or in a human-readable textfile format for easy import into other systems such as Matlab.

Semantic indexes can easily be built for relatively large corpora (up to some millions of documents or billions of words) on a typical laptop or desktop computer today.

Semantic Search. The basic method for semantic search proceeds first by creating a query vector by reading term or document vectors, and then scanning over the vectors and selecting those that score best compared with this query expression. The VectorStore and VectorSearcher use abstract interfaces so that developers can easily create alternatives with different I/O behavior, different query analysis logic, and different scoring functions. For I/O, this design has enabled in-memory search to support optimized server performance and large batch experiments. For query construction and scoring, it has enabled a full implementation of search using quantum logical connectives as pioneered by the Infomap project [7, Ch 7], holographic representations as implemented in the BEAGLE model [17], and permutation search [18].

Configuration and Extension. To encourage researchers and developers to use the package for new experiments, the design has consciously promoted easy configuration and extension where possible. Some of these decisions (reuse of off-the-shelf language analyzers from Lucene, and abstract VectorStore and VectorSearcher interfaces) have been mentioned already. Another example is the support for command line configuration using a dedicated Flags class. This uses Java's reflection operations to enable developers to create new global variables, document their intended usage, have their values parsed automatically from command line arguments, and use these variables anywhere in the codebase. This is a two-edged sword: enabling an external client to modify internal state from without is a risky violation of software encapsulation, but giving developers the power to easily control new internal functions without navigating the package's call stack has reduced the difficulty of learning the codebase well enough to prototype new algorithms.

Documentation and Discussion Forums. Documentation is crucial to a successful open source project at many levels, and we use many online tools to help with this. Firstly, all classes are documented internally using comments, and some of these comments become public HTML pages through the Javadoc system. Secondly, the project website at code.google.com also provides Wiki and bugtracking, and allows users to download an example corpus (the King James Bible). The bugtracking system is also used for feature requests, and the Wiki is used for a wide range of documentation including installation instructions, example querying commands and expected results using the example corpus, links to related research papers, and a comprehensive release log. Thirdly, there is also a SemanticVectors Google Group, which provides mailing lists and a web forum where questions are posted, experiments are proposed, and early results are often shared. Providing a quick, public response to questions and problems posted to the forum and bugtracking system is extremely important: this enables new users to surmount any difficulties quickly, and provides a great confidence boost to potential users who need to be assured that using the Semantic Vectors package will be easy and fruitful, not a frustrating timesink.

Testing. Maintaining a sophisticated codebase without automatic tests is almost impossible. The Semantic Vectors package uses the JUnit testing library from junit.org for unit testing key components, and for more complicated regression-style tests, which simulate the building and searching of semantic vector models and try to emulate the environment in which users actually find themselves as closely as possible.

Many other factors have gone into the success and growth of the project, but we believe that those noted above are especially important.

IV. Successful Research and Development Enabled by the Package

One testimony to the success of SemanticVectors is the wealth of algorithms and techniques that developers have coded up and contributed to the package. Some of these have existed for several years, others are recent — during the past few years, theoretical understanding of hyperdimensional vector calculus has progressed extremely rapidly and rigourously [19], and there are several other algebraic and geometric operations that are well-understood and still pending implementation. In addition, completely new algorithms have been successfully developed using SemanticVectors itself. This section will describe a summary of the existing algorithms, and will then give slightly more detail on the new research.

A. Implementations of Prior Research

This section briefly summarizes existing techniques that have been incorporated into the SemanticVectors package for general use.

Bilingual Models. Distributional models have been successfully adapted to multilingual applications using parallel corpora. The basic notion is that terms and documents from more than one language can be represented in the same vector space provided that some of the underlying elemental vector axes are the same for all languages: and such common elements can be obtained using translated documents as basic contextual units [7, §6.6]. Tools for building bilingual distributional models are a standard part of the SemanticVectors package, and have been extensively tested using the Europarl parallel corpus [20].

Clustering and Visualization. Clustering and visualization tools can help to get a sense of the way words are related to one another beyond what can be understood from scanning a ranked list of results. SemanticVectors ships with a simple kMeans clustering algorithm built in, and with tools for producing 2-dimensional pictures of local regions of the vector space using a projection algorithm based upon singular value decomposition [7, §6.4].

Permutation and Convolution. One of the most persistent charges levelled against traditional vector space models for distributional semantics is that they are "bag of words" representations — that is, they fail to take word order into account. However, there are many tools in vector algebra that can be used to model effects that depend upon ordered sequences. Two such methods that have met with success recently are permutation and convolution.

Permutation indexing, developed successfully by [18], uses a permutation of coordinates to model the transition from one word to the next. Thus if Π is a permutation, and the word a is always followed by the word b and the word b is always preceded by the word a, we would expect that $\Pi(a) = b$ (where by abuse of notation a is used both for a word and the corresponding word vector). This technique was successfully (and quite easily) incorporated into the SemanticVectors package, with immediately impressive results. For example, in the King James Bible model, the query *'king ?'* returns names of individual Biblical kings, whereas the query *'king of ?'* returns names of Biblical cities and countries.

Convolution is another technique for incorporating word order, using a non-commutative *convolution product* $a \cdot b$ which is a projection of the tensor product into a lower-dimensional space that avoids the exponential space cost that repeated use of the tensor product incurs [22]. Convolution methods for encoding word order effects were used successfully in the BEAGLE model of [17], and have also been incorporated into SemanticVectors.

B. Predicate methods

Predication-based Semantic Indexing (PSI) [23] adapts the permutation-based method introduced by Sahlgren and his colleagues to encode structured medical knowledge. Rather than encoding the relative position of terms, PSI uses permutations to encode the predicate type connecting two arguments in an object-relation-object triplet (such as "sherry IS_A wine"), or predication. In addition, rather than encoding relations between terms, the encoded predications occur between discrete concepts in the Unified Medical Language System (UMLS) meta-thesaurus [24]. These predications are extracted by the SemRep system [25] from the biomedical literature. The procedure to encode and retrieve predications into a vector space is analogous to the procedure used for positional indexing in Semantic Vectors. However, rather than shifting the position of vector elements based on the relative position of terms, an index number is assigned to each of the 40 predicate types (such as TREATS or CAUSES) encoded by

TABLE III
PERMUTATION AND SIMILARITY RESULTS FOR GEOPOLITICAL ENTITIES [21]. NOTE THE WAY IN WHICH THE PERMUTATION QUERY REPRESENTS THE TARGETS OF A SEMANTIC RELATION MUCH MORE ACCURATELY THAN TRADITIONAL SIMILARITY QUERIES.

Permutation query *“king of ?”*	
0.728	assyria
0.699	babylon
0.662	syria
0.647	zobah
0.604	persia
0.532	judah
0.556	jarmuth
0.542	ellasar
0.536	bashan
0.533	belial

Permutation query *“king ?”*	
0.712	ahasuerus
0.502	agrippa
0.502	dale
0.448	chamberlain
0.432	ahaz
0.426	arising
0.411	cupbearer
0.409	arad
0.408	solomon
0.397	jareb

Similarity query *“king”*	
1.000	king
0.450	reign
0.449	over
0.442	all
0.438	so
0.412	servants
0.405	did
0.397	judah
0.389	now
0.381	queen

Similarity query *“assyria”*	
1.000	assyria
0.768	sepharvaim
0.754	gozan
0.717	hena
0.717	ivah
0.716	trustest
0.688	rabshakeh
0.683	hezekiah
0.659	shalmaneser
0.653	shebna

SemRep. Consequently, it is possible to encode and retrieve the type of predicate connecting two UMLS concepts, rather than the relative position in which two terms are likely to occur. Table IV, below, shows the results of some predication-based searches on a 500-dimensional PSI space derived from a set of over 20 million predications extracted by SemRep from titles and abstracts added to the biomedical literature over the past decade.

When combined with Pathfinder network scaling [26], PSI is able to derive plausible chains of reasoning linking two concepts. In the example in Figure 1, a PSI-based associative network is used to link the concepts “beer” and “wine”, by finding the five nearest neighbors of a composite vector representing both of these concepts, and preserving the most significant links using Pathfinder. (Note that Figure 1 shows automatic, not hand-curated results, the fact that “alcoholic beverages” are “beverages” has been missed, and “beer” is linked to both of these concepts instead of factoring through “alcoholic beverages” alone.) This figure was produced using the EpiphaNet system (http://epiphanet.uth.tmc.edu), which is designed to allow researchers to explore concepts of interest from the biomedical literature, with encouraging results [27]. The Semantic Vectors project served as a catalyst for the development of the PSI model. The ideas underlying PSI emerged during work on permutation-based indexing in Semantic Vectors, which in turn provided an easily extensible platform for the rapid prototyping of PSI.

C. *Reflective Random Indexing*

Of the methods of distributional semantics, the scalability offered by Random Indexing makes it particularly appealing in the biomedical context, on account of the rapid growth of literature in this domain. One application in this area is literature-based discovery, which seeks to derive previously unrecognized and therapeutically useful connections between biomedical concepts. One way of approaching this problem is to find meaningful connections between terms that do not co-occur directly in any document, a facility of certain distributional models that has been termed “indirect inference” [9]. However, Random Indexing in its original implementation does not address the issue of indirect inference, as the resulting matrix is a reduced-dimensional approximation of the original term-by-document matrix, in which each document is represented as an independent dimension. In this original matrix, the vectors for two terms that do not have a document in common will share no non-zero dimensions, and consequently their relatedness as measured with the commonly used cosine metric will be zero. As predicted by the Johnson-Lindenstrauss Lemma [12], Random Indexing preserves the distances between vectors in the original term-document matrix with high probability, limiting its usefulness as a means of deriving indirect inference. A solution to this problem emerged from the possibility of cyclical retraining which was implemented as an experimental feature in Semantic Vectors. The process of generating document vectors from term vectors, and in turn term vectors from document vectors can be performed iteratively. We have called this process Reflective Random Indexing (RRI), and it has been shown to improve the ability of Random Indexing to derive meaningful indirect inferences [13]. Sample results obtained using Reflective Random Indexing are shown in Table V.

When evaluated for the ability to predict terms that would co-occur directly in the future from a time-delimited segment of the biomedical literature, reflective variants outperformed both the original and sliding-window based implementations of Random Indexing, in a study which compared the overall percentage of the 50 nearest-indirect neighbors of 2000 randomly selected biomedical terms from all of the citations added to MEDLINE between 1980 and 1985 that co-occurred directly in some abstract added after this time period. The ability of Random Indexing to predict future co-occurrence improves with up to two cycles of reflection. However repeated iterations lead to a decrease in performance as term vectors tend to converge and can no longer be discriminated from one another. The ability to obtain meaningful indirect inference at minimal computational cost has implications for information retrieval in general, as this allows for the retrieval of documents that are related to a query cue term, but do not contain this specific term — an original motivation for the development of Latent Semantic Indexing (LSI) [8].

As was the case with PSI, an adaptation of code contributed to the Semantic Vectors platform was used to rapidly prototype

TABLE IV

PREDICATION-BASED RETRIEVAL. THE ANSWERS TO SIMPLE CLINICAL QUESTIONS SUCH AS "WHAT TREATS DEPRESSIVE DISORDER?" CAN BE RETRIEVED FROM THE PSI SPACE (LEFT). ALTERNATIVELY, IT IS POSSIBLE TO OBTAIN THE STRONGEST RELATIONS ACROSS ALL PREDICATE TYPES (RIGHT). THE RESULTS IN ALL CASES ARE PLAUSIBLE AND CAN BE LINKED BACK TO SUPPORTING EVIDENCE IN THE LITERATURE.

? TREATS depressive disorder	**? ANY PREDICATE depressive disorder**
0.98 : lexapro	1.0:abnormal cortisol $\rightarrow$ PREDISPOSES
0.77 : problem solving therapy	1.0:body weight problem $\rightarrow$ PREDISPOSES
0.74 : sertralin	1.0:unpleasant memories $\rightarrow$ PREDISPOSES
0.68 : marital therapy	1.0:conditioned helplessness $\rightarrow$ AFFECTS
0.66 : stablon	1.0:depressed parent $\rightarrow$ PREDISPOSES

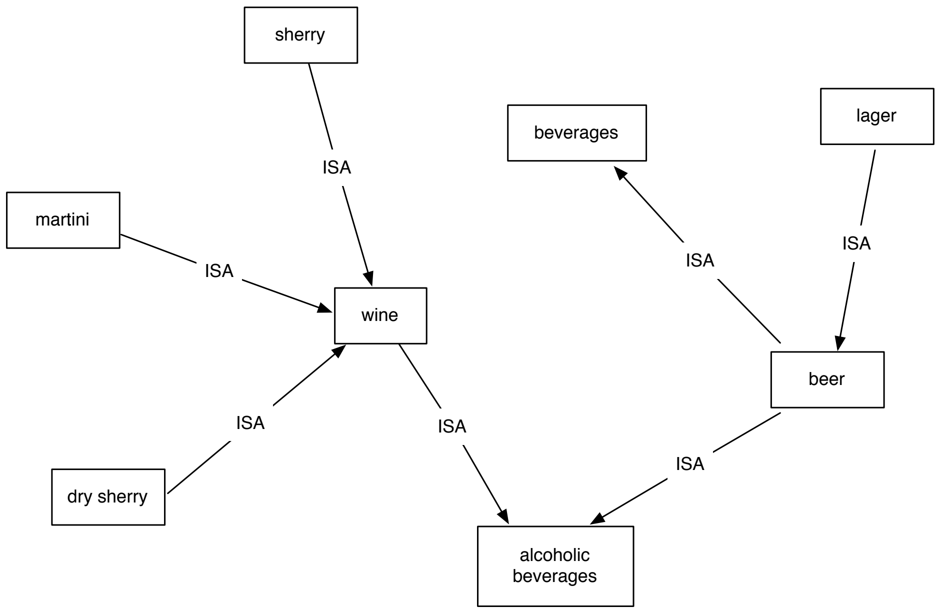

Fig. 1. Extracted ISA relationships between different beverage concepts

TABLE V

REFLECTION AND INDIRECT INFERENCE. THE NEAREST INDIRECT NEIGHBORS (THE MOST STRONGLY RELATED TERMS THAT DO NOT CO-OCCUR DIRECTLY IN ANY DOCUMENT) OF THE TERMS "COMPUTATIONAL" AND "SEMANTICS" AS DERIVED FROM THE TOUCHSTONE APPLIED SCIENCES (TASA) CORPUS USING RRI (LEFT), AS COMPARED WITH RI AS ORIGINALLY IMPLEMENTED (RIGHT).

Reflective Random Indexing		**Random Indexing**	
Semantics	Computational	Semantics	Computational
0.30 syntactic	0.32 primary	0.19 schoolers	0.18 assistive
0.30 meanings	0.31 microcomputers	0.18 ohh	0.18 redfields
0.27 grammatical	0.31 minicomputers	0.18 lei	0.18 dlm
0.27 comprehend	0.30 technological	0.17 kula	0.17 bearsbird
0.27 phrases	0.29 capability	0.17 felicia	0.17 mensural

RRI, both for initial investigations into indirect inference [13], and for a later study that successfully applied RRI to the problem of automated indexing of the biomedical literature [28]. In addition, the idea that iteration may provide a solution to the limited ability of RI to derive indirect inferences emerged from a discussion on the Semantic Vectors online forum.

D. Other Novel Uses

This section briefly summarizes some more of the novel projects for which SemanticVectors has been used as an off-the-shelf component.

Technology Management. The SemanticVectors project was first commissioned and supported by the University of Pittsburgh as part of an online technology management search engine described in [15].

Scientific Article Analysis. The package has been used as part of a semantic analysis system that visually describes the overlap between different scientific disciplines, created by semantic mapping of journal articles [29].

Other applications described in the project online documentation include: integration with LeActiveMath for a tutorial system; distributional studies of social networking; word sense discrimination and disambiguation; alignment of knowledge resources such as Wikipedia and WordNet; and language games in artificial intelligence. A more dynamic and complete set of references is available on the project Wiki.

V. Conclusions

Distributional semantics, and vector models in particular, have demonstrated useful results in many experiments, so much so that industrial applications should be more well-developed than they have been to date. We believe that part of the reason for this lag between research and implementation has been due to the lack of scalable and reliable software, a gap we are trying to fill with the SemanticVectors package.

The SemanticVectors package is growing increasingly useful and popular and has been used in a variety of scientific and engineering projects. From an engineering standpoint, the open nature of the package's development, the simple architecture of the main systems, and the wealth of tools available for collaborating online, have helped to make the platform stable and usable. This enables researchers to focus on creating new applications, and the wealth of mathematical techniques and application domains available has led the package to be used in many innovative projects.

References

[1] L. Wittgenstein, *Philosophical Investigations*. Blackwell: Blackwell, 1953, 3rd edition, 2001.

[2] J. Firth, "A synopsis of linguistic theory 1930-1955," *Studies in Linguistic Analysis, Philological Society, Oxford*, reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow. 1957.

[3] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan Kaufmann, 1999.

[4] K. Sparck Jones, *Synonymy and Semantic Classification*. Edinburgh University Press, 1986, (Originally Cambridge PhD thesis, 1964).

[5] G. Salton and M. McGill, *Introduction to modern information retrieval*. New York, NY: McGraw-Hill, 1983.

[6] C. van Rijsbergen, *The Geometry of Information Retrieval*. Cambridge University Press, 2004.

[7] D. Widdows, *Geometry and Meaning*. Stanford, California: CSLI publications, 2004.

[8] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41(6), pp. 391–407, 1990.

[9] T. Landauer and S. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[10] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. S.I.A.M., 1997.

[11] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2001, pp. 245–250.

[12] W. Johnson and J. Lindenstrauss, "Extension of lipshitz mapping to hilbert space," *Contemporary Math*, vol. 26, pp. 189–206, 1984.

[13] T. Cohen, R. Schvaneveldt, and D. Widdows, "Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections," *Journal of Biomedical Informatics*, 2009.

[14] M. Sahlgren, "The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces," Ph.D. dissertation, Department of Linguistics, Stockholm University, 2006.

[15] D. Widdows and K. Ferraro, "Semantic vectors: A scalable open source package and online technology management application," in *Proceedings of the sixth international conference on Language Resources and Evaluation (LREC 2008)*, Marrakesh, Morroco, 2008.

[16] P. Kanerva, J. Kristofersson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, 2000.

[17] M. N. Jones and D. J. K. Mewhort, "Representing word meaning and word information in a composite holographic lexicon," *Psych. Review*, vol. 114, no. 1, 2007.

[18] M. Sahlgren, A. Holst, and P. Kanerva, "Permutations as a means to encode order in word space," in *Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci'08)*, Washington D.C., 2008.

[19] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, June 2009.

[20] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MY Summit*, 2005.

[21] D. Widdows and T. Cohen, "Semantic vector combinations and the synoptic gospels," in *Proceedings of the Third International Symposium on Quantum Interaction*, Saarbrücken, March 2009.

[22] T. Plate, *Holographic Reduced Representations: Distributed Representation for Cognitive Structures*. CSLI Publications, 2003.

[23] T. Cohen, R. Schvaneveldt, and T. Rindflesch, "Predication-based semantic indexing: Permutations as a means to encode predications in semantic space," in *Proceedings of the AMIA annual symposium*, San Francisco, 2009.

[24] A. McCray, A. Aronson, A. Browne, T. Rindflesch, A. Razi, and S. Srinivasan, "UMLS knowledge for biomedical language processing," *Bulletin of the Medical Library Association*, vol. 81, p. 184, 1993.

[25] T. Rindflesch and M. Fiszman, "The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text," *Journal of Biomedical Informatics*, vol. 36, pp. 462–477, 2003.

[26] R. Schvaneveldt, *Pathfinder associative networks: studies in knowledge organization*. Norwoood, NJ, USA: Ablex Publishing Corp, 1990.

[27] R. Schvaneveldt, T. Cohen, and K. Whitfield, "Paths to discovery," in *Proceedings of the 36th Carnegie Mellon Symposium on Cognition*, Pittsburgh, PA, USA, June 2009.

[28] V. Vasuki and T. Cohen, "Reflective random indexing for semi-automated indexing of medline abstracts," in *Proceedings of the AMIA Symposium*, 2009.

[29] G. Newton, A. Callahan, and M. Dumontier, "Semantic journal mapping for search visualization in a large scale article digital library," in *Second Workshop on Very Large Digital Libraries at the European Conference on Digital Libraries (ECDL)*, 2009.