The Semantic Librarian: A search engine built from vector-space models of semantics

Harinder Aujla¹ • Matthew J. C. Crump² • Matthew T. Cook³ • Randall K. Jamieson³

© The Psychonomic Society, Inc. 2019

Abstract

Check for updates

Psychologists have made substantial progress at developing empirically validated formal expressions of how people perceive, learn, remember, think, and know. In this article, we present an academic search engine for cognitive psychology that leverages computational expressions of human cognition (vector-space models of semantics) to represent and find articles in the psychological record. The method shows how psychological theory can be used to inform and aid the design of psychologically intuitive computer interfaces.

Keywords Cognitive computing · Document representation and retrieval · Search engine · BEAGLE · Computational linguistics

Traditionally, artificial intelligence technologies have been developed by engineers, computing scientists, and statisticians. However, cognitive psychology—a discipline dedicated to developing rigorous accounts of how people perceive, learn, remember, think, and know—is in a good position to contribute to the larger endeavor.

A number of forward-looking psychologists have already applied formal cognitive theories to a variety of problems. Johns et al. (2018) used a model of human semantic memory to predict mild cognitive impairment from verbal behavior. T. Rubin, Koyejo, Jones, and Yarkoni (2016) used semantic models to summarize imaging data from the psychological record (see also T. N. Rubin et al., 2017). Kwantes, Derbentseva, Lam. Vartanian, and Marmurek (2016) used semantic models to predict personality profiles from essay data. Bedi et al. (2015) used semantic models to predict mental health from verbal reports (see also Cook, 2018). Foltz, Laham, and Landauer (1999) used a semantic model to grade undergraduate essays. Brosowsky and Crump (2018) used semantic modeling to detect lying in typed stories. Graesser (2011; see also Nye, Graesser, & Hu, 2014) and McNamara (e.g., Roscoe et al., 2014; Roscoe & McNamara, 2013) have developed psychologically informed tutoring

Harinder Aujla h.aujla@uwinnipeg.ca

- ¹ University of Winnipeg, Winnipeg, Manitoba, Canada
- ² Brooklyn College, New York, NY, USA
- ³ University of Manitoba, Winnipeg, Manitoba, Canada

systems. Brooks's (1991) work on cognitive subsumption architectures has advanced robotics. And, of course, artificial neural networks have long served as an engine in intelligent systems (e.g., LeCun, Bengio, & Hinton, 2015; Rosenblatt, 1958; Rumelhart, Hinton, & McClelland, 1986).

The work presented here follows in this tradition by applying tools and methods from computational cognitive psychology to present a cognitive search engine. Because our method is constructed using psychological theories of natural language processing, it acts as a principled *cognitive surrogate* that interprets a user's query and returns a list of documents that fit with the user's intent.

The problem

Scientists and scholars often depend on keyword search engines to retrieve information. Although keyword matching works well in some cases, the technique suffers several shortcomings. First, keyword matching assumes a simple relationship between a signifier (i.e., a word) and its signified (i.e., the word's meaning; de Saussure, 2011). But, that premise is naïve: words can have multiple meanings (e.g., *bank*) and a word's meaning can change depending on the context in which it appears (e.g., a *rough* draft versus a *rough* ride). As a consequence, keyword matching can lead to problems in finding documents that use different words to express the same idea or misconstrue relationships between documents that use the same words to express different ideas. Second, different research traditions use different vocabularies. For example, linguists and psycholinguists sometimes use different words to discuss the same problems. Consequently, where meaning overlaps but vocabulary differs, keyword search can be blind to those connections. Third, keyword search assumes the user's vocabulary, but if a user does not already have the vocabulary needed to conduct a search, they find themselves in a catch 22.

Information scientists have tried to solve the problem by developing methods for indexing documents by meaning (Bontcheva, Tablan, & Cunningham, 2014). One dominant method is semantic annotation—a method in which humans or machines add semantic metadata to the documents in a database. The method can be effective: once documents are semantically annotated, a user can search for documents by matching to the semantic metadata. However, those methods still depend upon keyword search and suffer from the shortcomings already noted. So, then, how do we improve search to make it more intuitive so that the user is interpreted as intended and search results are thusly optimized?

Vector-space models of semantics

Psychologists have worked since the 1940s to derive a quantitative representation of word meaning. Osgood's (1952) work on the semantic differential stands as the seminal contribution to that effort. In the 1960s and 1970s, focus shifted to deriving a hierarchical representation of word meaning in propositional networks (e.g., Anderson, 2013; Collins & Loftus, 1975; Collins & Quillian, 1969). In the 1980s, the strategy shifted again toward the quantification of word meaning based on people's introspective ratings about word properties (Friendly, Franklin, Hoffman, & Rubin, 1982; Gilhooly & Logie, 1980; D. C. Rubin & Friendly, 1986; Toglia & Battig, 1978). All of these methods laid the foundations for a psychological theory of language and knowledge. However, they relied on introspective judgements that were experimentally expensive to obtain, thereby limiting the scope of experimental efficiency.

In the 1990s, the psychological analysis of semantics leapt forward with the development of new corpus-based vectorspace models. In contrast to prior methods, the vector-space models leveraged computational methods to arrive at an efficient, broad, and deep analysis of word meaning based on patterns of word use in printed text (e.g., newspapers and encyclopedias).

Naturally, vector-space models differ by formulation, but they all aspire to a common goal: to represent the semantic relationships between words in a high-dimensional geometry in which words that share meaning are located in similar regions of semantic space (i.e., typically measured by cosine similarity). They also share a method for assessment, in which success is evaluated against experimental data to ensure a correspondence between people's and the model's understanding. Latent semantic analysis (LSA) is a first-generation vectorspace model of meaning (Landauer & Dumais, 1997). According to LSA, word meaning is derived from a text corpus by tabulating word co-occurrence in a word-by-document matrix, transforming the counts in the word-by-document matrix into corresponding measurements of entropy, decomposing the transformed word-by-document matrix using singular value decomposition, and re-constructing the matrix in a reduced dimensionality. In the end, each word is represented by a unique semantic vector. Despite LSA's simplicity, it tracks human language behavior, including the rate of language acquisition, human vocabulary judgments, word-sorting behavior, free association behavior, and categorization.

Bound Encoding of the Aggregate Language Environment (BEAGLE) is a second-generation semantic model (Jones, Kintsch, & Mewhort, 2006; Jones & Mewhort, 2007). BEAGLE operates by deriving a semantic vector for each word in a text corpus based on principles of holographic reduced representation (Plate, 1995), and improves over LSA in several ways. First, BEAGLE derives a representation of word meaning that is conditional on word order; LSA does not. Second, BEAGLE encodes information related to syntax and grammar; LSA does not. Third, BEAGLE outperforms LSA in both its scope and precision at tracking human language behavior. Fourth, BEAGLE is grounded in established principles of human memory theory, and therefore makes contact with a history of theoretical and empirical advances in human cognition (Murdock, 1982, 1983, 1995, 1997).

Building on BEAGLE, BEAGLE Random Permutation (BEAGLE-RP; Sahlgren, Holst, & Kanerva, 2008) is a more recent expression of its parent theory that adopts features of Kanerva's (1994) spatter code model. In contrast to BEAGLE, BEAGLE-RP uses sparse representation with ternary vectors instead of real-valued numbers and incorporates index overlap rather than holographic methods to compute associations. The use of index overlap renders the theory consistent with Hebbian learning and long-term potentiation (i.e., neurons that "fire together wire together"). Recent findings suggest that semantic modeling of human behavior with BEAGLE-RP vectors and, on the practical side, yields advantages with respect to scalability (Recchia, Sahlgren, Kanerva, & Jones, 2015)

Although LSA, BEAGLE, and BEAGLE-RP differ in important ways, they converge on the common goal of deriving a psychologically valid vector-space representation of word meaning. To the extent that they predict human language judgments, all three theories offer a sound psychologically and empirically informed base representation of word meaning. But, is that base representation sufficient to support the construction of a useful and psychologically valid search engine?

The work that follows uses BEAGLE and BEAGLE-RP as the underlying methods for developing a semantically indexed search engine—the "Semantic Librarian." In a first step, we apply the methods to derive word meanings from publications in the field of experimental psychology. In a second step, we use the word vectors to derive a representation for each document in that record. In a third step, we evaluate the system in a set of objective tests that demonstrate that the engine can recover a document in noise. In a fourth step, we describe and articulate a working web interface that scientists can use to search the psychological record.

The Semantic Librarian

Representation

The corpus

To derive the semantic word vectors, we need a corpus. To develop the corpus, we scraped data from 27,560 documents (i.e., titles, abstracts, author names, and keywords) published in experimental psychology journals, including the *Canadian Journal of Experimental Psychology*: General (1916–2015), Journal of Experimental Psychology: General (1916–2015), Journal of (1975–2016), Journal of Experimental Psychology: Animal Learning and Cognition (1975–2016), Journal of Experimental Psychology: Human Perception and Performance (1975–2016), Journal of Experimental Psychology: Learning, Memory, and Cognition (1975–2016), and Psychological Review (1894–2016).

Deriving semantic vectors with BEAGLE

Next we applied the BEAGLE method so as to derive a semantic memory vector for each word in the corpus.

Broadly, BEAGLE works by "reading" a text corpus and, on route, encoding a memory vector that represents the meaning of each word in that corpus. Mechanistically, the model is expressed in algebra.

At the outset of a simulation, each of the *i* unique words in the corpus is represented by a randomly generated environmental vector, e_i . Each environment vector has dimensionality *n* and each element in an environment vector takes a value randomly sampled from a normal distribution with mean zero and variance 1/n. In the simulations that follow, and consistent with tradition, dimensionality was set to n = 1,024. Although the semantic memory vector for each word changes as the model reads the corpus, the environment vectors are stable over a simulation; serving as unique identifiers for the words in the corpus (i.e., each word's orthographic and phonological identity).

Next, the model "reads" the corpus one sentence at a time to build a semantic memory vector for each word, m_i . The semantic memory vector for each word is composed of two kinds of information: context information and order information.

Context information is computed by summing the environmental vectors for all other words in the same sentence (i.e., excluding the word of interest)

$$m_i = m_i + \sum_{j=1}^{j=\lambda} e_j, \quad \text{where } i \neq j;$$
 (1)

here, m_i is the semantic memory vector for word *i* in the sentence, e_i is the environment vector for word *i* in the sentence, and λ is the number of words in the sentence. For example, after reading the sentence, "A dog bit the mailman," the memory vector for *dog* is updated as $m_{dog} = m_{dog} + e_{bit} + e_{mailman}$, the memory vector for *bit* is updated as $m_{bit} = m_{bit} + e_{dog} + e_{mailman}$, and the memory vector for *mailman* is updated as $m_{mailman} = m_{mailman} + e_{dog} + e_{bit}$. Note that all of words in the sentence are not included in the construction of the context representation. The words that are excluded because they occur so often in text that including them forces all words to become unrealistically similar to one another.

Summing the environment vectors in this manner causes the memory vectors for all words that co-occur in the same sentence to grow similar to one another; because they are composed of the same environment vectors. However, the method also encodes indirect (i.e., higher-order) associations between words. This happens because words with shared meaning co-occur with the same words, even if the words with shared meaning never co-occur in the same sentence. For example, even if *dog* and *beagle* do not co-occur in the same sentence in the corpus, they become similar to one another by virtue of having common words summed into their representations (e.g., *loyal* and *vicious*).

Order information is computed by encoding information about which words follow one another in a sentence and updating the memory vector with that information. In particular, the first-order association between words (i.e., immediately adjacent words) is encoded using noncommutative circular convolution; hereafter denoted as circular convolution.

Circular convolution is a vector operation that binds two vectors, \mathbf{x} and \mathbf{y} , to produce an associative vector, \mathbf{z} :

$$z_i = \sum_{j=0}^{n-1} x_{jmodn} \times y_{(i-j)modn} \quad \{ \text{for } i = 0 \text{ to } n-1 \},$$
(2)

where, n is the dimensionality of **x** and **y** and the vectors **x** and **y** are indexed by modulo subscripts. A convenient property of circular convolution is that it produces a vector **z** that is the same dimensionality as the inputs **x** and **y**, thereby allowing the association between **x** and **y** to be summed into a single vector along with the context information.

Higher-order sequential information in a sentence (e.g., sequences of three, four, or more words) is computed by applying circular convolution recursively and incorporating those computations into the word's semantic vector,

$$m_{i} = m_{i} + \sum_{j=1}^{j=p\lambda - (p^{2}-p)-1} bind_{ij},$$
(3)

where, m_i is the memory vector for word *i*, *p* is the position of word *i* in the sentence, and $bind_{ij}$ is the *jth* convolution for the word being coded.

To illustrate the operation, the order information for the word dog, m_{dog} , in the sentence, "a dog bit the mailman," is encoded as a sum of the following:

$$\begin{array}{l} bind_{dog,1} = e_{a} \circledast \Phi \\ bind_{dog,2} = \Phi \circledast e_{bit} \\ bind_{dog,2} = \Phi \circledast e_{bit} \\ bind_{dog,3} = e_{a} \circledast \Phi \circledast e_{bit} \\ bind_{dog,4} = \Phi \circledast e_{bit} \circledast e_{the} \\ \end{array} \right\} \text{Trigrams}$$

$$\begin{array}{l} bind_{dog,5} = e_{a} \circledast \Phi \circledast e_{bit} \circledast e_{the} \\ bind_{dog,6} = \Phi \circledast e_{bit} \circledast e_{the} \circledast e_{mailman} \\ \end{array} \right\} \text{Quadgrams}$$

$$\begin{array}{l} bind_{dog,7} = e_{a} \circledast \Phi \circledast e_{bit} \circledast e_{the} \circledast e_{mailman} \\ \end{array} \right\} \text{Quintagram},$$

where denotes circular convolution and Φ is a constant and universal placeholder (i.e., a unique environment vector) used in the computation of order information and that is the same for every word in every position in every sentence.

Taken together, a word's meaning is equal to the sum of its context and order information after the model has "read" an entire text corpus,

$$m_{i} = m_{i} + \sum_{j=1}^{j=\lambda} e_{j} + \sum_{j=1}^{j=p\lambda - (p^{2}-p)-1} bind_{ij},$$
(5)

where m_i is the semantic memory vector for word i, e_j is the environment vector for word j in the sentence, λ is the number of words in the sentence, and p is the position of word i in the sentence.

In summary, BEAGLE uses the environment vectors to develop semantic memory vectors that represent the meaning of each word in the corpus as a combination of both its context and order information. As the algebra indicates, the theory predicts that a word's meaning will reflect its history of cooccurrence with, and position relative to, other words in sentences. Thus, BEAGLE implements the wisdom from linguistics that "You shall know a word by the company it keeps" (Firth, 1957).

Deriving semantic vectors with BEAGLE-RP

BEAGLE-RP is theoretically consistent with BEAGLE. However, it applies a different method to build the semantic vectors (see Recchia et al., 2015; Sahlgren et al., 2008).

First, each unique word in the corpus is represented by an environmental vector of 3,000 dimensions, with 30 values of + 1 and 30 of - 1 assigned at random to its elements (all other elements equal to zero).

Context vectors are defined in a similar fashion to how they are in BEAGLE:

$$m_i = m_i + \bigvee_{j=1}^{j=\lambda} e_j \quad \text{where } i \neq j; \tag{6}$$

here, m_i is the memory vector for word *i*, V is an index operator (more on this below), λ is the number of words in the sentence, and e_i is the environment vector for word *j* in the sentence.

The V index operator computes a vector of the same dimensionality as the inputs, such that element *j* is equal to element *j* in the input vectors if at least two of the words in the sentence share the same value (i.e., +1 or -1). Figure 1 presents an example of computing the context vector for the word "dog" in the sentence "a dog bit the mailman." As is shown, both e_{bit} and $e_{mailman}$ share a value of -1 in the fifth index, and therefore the sentence vector includes a single non-zero value (i.e., -1 in index 5).

As is indicated in Eq. 6, the context vector is added to the word's memory vector. Thus, whereas $\bigvee_{j=1}^{j=\lambda} e_j$ contains only the values + 1, - 1, and 0 in its elements, each element in a memory vector, m_i , can contain any value between -w and +w, where w is the number of times that word i appeared in the corpus.

The BEAGLE-RP model captures order information by nearly the same method as it captures context information. However, the order information for a word at position p in a sentence is computed on the basis of the words that appear in positions p - 2, p - 1, p + 1, and p + 2 in the same sentence. In addition, the environment vectors for the words in positions p- 2, p - 1, p + 1, and p + 2 are changed depending on their serial position—the operation that identifies each unique word as a function of its position in a sentence. Thus, order information in BEAGLE-RP is computed as

$$m_i = m_i + \bigvee_{j=p-2}^{j=p+2} e_j, \quad \text{where } i \neq j \text{ and } 0 < j \le \lambda, \tag{7}$$

 m_i is the memory vector for word *i*, V is the index operator already described, e_i is the environment vector for the word at position *j* in the sentence following permutation based on its position relative to p, and λ is the length of the sentence. By tradition, and for the sake of convenience, the position vectors in our simulations were generated by shifting the indices of the original environmental vector in position p-2 two places to the left, in position p-1 one place to the left, in position p+1one place to the right, and in position p + 2 two places to the right, with the values for indices ≤ 0 or $\geq 3,000$ wrapped around to the end or start of the vector, respectively. Finally, we deviated from the procedure described by Recchia et al. (2015) with respect to the treatment of sentence boundaries. Whereas Recchia et al. incorporated order information with a $p \pm 2$ window around the target word, irrespective of sentence position, we only incorporated order information within $a \pm 2$ word boundary within a sentence. For example, only the order information for words preceding the last word in a sentence

	Index										
	1	2	3	4	5		2996	2997	2998	2999	3000
$e_a =$	0	0	0	0	0		0	0	0	0	0
e _{dog} =	_		_	_	_		_	_	_	_	_
e _{bit} =	0	1	0	0	-1	• • •	0	0	0	-1	0
<i>e</i> _{the} =	0	0	0	0	0		0	0	0	0	0
e _{mailman} =	0	0	0	0	-1		0	0	0	0	1
	ļ	ļ	ļ	ļ	ļ		ļ	ļ	ļ	ļ	ļ
	0	0	0	0	-1		0	0	0	0	0

Fig. 1 An example of computing the context vector for the word *dog* in the sentence "a dog bit the mailman"

were included in its order vector.

In summary, BEAGLE-RP uses the environment vectors to construct semantic memory vectors that represent the meaning of each word in the corpus as a combination of both its context and order information. As the algebra indicates, the theory, like BEAGLE, predicts that a word's meaning will reflect its history of co-occurrence with, and position relative to, other words in sentences.

Building the document vectors

Once we had derived the semantic vectors for all 40,517 words in the 27,560 documents in the journal corpus using both the BEAGLE and BEAGLE-RP methods, we used the word vectors to construct representations for each of the 27,560 documents.

Each document vector was computed as the sum of the semantic memory vectors that corresponded to all w words in the document's title, abstract, and keyword list:

$$d_i = \sum_{j=1}^{j=w} m_j, \tag{8}$$

where d_i is the semantic summary of document *i*, m_j is the semantic memory vector corresponding to word *j* in document *i*, and *w* is the number of words in document *i*. Once constructed, the document representation was stored to the database of 27,560 documents.

Searching the document space

To search the document space, we constructed a query vector, **q**, that was equal to the sum of the semantic memory vectors

corresponding to all *w* words in the query:

$$q = \sum_{j=1}^{j=w} m_j, \tag{9}$$

where q is the search query, m_j is the semantic memory vector for word j in the search query, and w is the number of words in the search query.

Once computed, the search query, \mathbf{q} , was used to search the database, and a ranked list of the documents was retrieved. The ranked list was constructed by, first, computing the cosine similarity between \mathbf{q} and the representation for each of the $i = 1 \dots 27,560$ document vectors in the database:

$$\operatorname{Sim}(q, d_i) = \frac{\sum_{j=1}^{j=n} q_j \times d_j}{\sqrt{\sum_{j=1}^{j=n} q_j^2} \sqrt{\sum_{j=1}^{j=n} d_j^2}},$$
(10)

where **q** is the vector representing the search term, d_i is the semantic vector summary of document *i*, and *n* is the dimensionality of the vectors under comparison. Once the similarity of the query to all documents was computed, a ranked list of the 27,560 documents was returned, so that the document most similar to **q** was returned first (i.e., rank = 1) and the document least similar to **q** was returned last (i.e., rank = 27,560). Thus, a document retrieved itself perfectly if it retrieved itself at rank = 1.

In summary, we derived vector representations of word meaning using BEAGLE and BEAGLE-RP, we used the word representations to encode documents, and we used a search query to retrieve a ranked list of all documents.

Assessing the system

At face value, our method presents a psychologically valid semantic search engine: It uses modern theories of human semantic memory to represent and retrieve documents. But, does it work? To evaluate the system, we developed a set of simple, verifiable tests that provide a rational basis for discriminating performance between models.

Simulation 1: Recovering a target document

In Simulation 1, we asked whether our method can recover a target document. To answer the question, we conducted a Monte Carlo study. In each simulation, we sampled a document from the journal database, randomly sampled a percentage of words from the document's abstract, title, and keywords; constructed a search query from that set of randomly sampled words; queried the database; and recorded the retrieval rank of the target document. To evaluate the system's loss tolerance, we conducted 1,000 simulations for queries composed of 5%, 10%, 25%, 50%, and 100% of the words from the document's abstract, title, and keywords.

To assess the model against rational controls, we conducted two additional sets of simulations. The first control method repeated the Monte Carlo simulation, but used random vectors rather than the semantic vectors to define word meaning (i.e., the environment rather than the semantic memory vectors). The second control method eschewed the vector-based method altogether and constructed the document list based on the number of times each word in the query occurred in a document. In those simulations, the document retrieved at rank = 1 had the largest word overlap with the words sampled from the target document.

Figure 2 shows the median retrieval ranks for the target document, depending on the percentage of words included in the query.

As is shown in Fig. 2, all four methods worked well, performing perfectly in the majority of simulations (i.e., median rank = 1) and retrieving the target document no worse than a median rank of 12 out of 27,560, even when only a very small percentage of words were sampled to the query. In fact,

as long as more than 25% of the words in the document were included in the query, all methods retrieved the target document at median rank = 1. We conclude that the semantic search method can recover a target document very well and that it is surprisingly tolerant to a noisy query. However, that conclusion was true of all methods and raises questions about the value of using semantic vectors at all.

Simulation 2: Comparison of semantic and nonsemantic vector methods

To expose the advantage of using semantic vectors, we reconducted Simulation 1, but we constructed the search query using semantic associates of the words from the target document (e.g., the word *memory* from the document was replaced by the word *storage* in the query). For this simulation, we asked how well a target document could be retrieved on the basis of a match to the *semantic relationship* rather than the *particular words* in a query. To conduct the simulation, we used semantic associates in the query that were nearest neighbors to the words sampled from the document (in the nonsemantic simulations, we used the environment vectors that corresponded to the nearest neighbors).

The results of the simulation are presented in Fig. 3; word match results are not presented because the method for replacing words with synonyms rendered the method entirely ineffective.

As is shown in Fig. 3, both semantic methods recovered the target document much better than the nonsemantic method. In fact, the nonsemantic method does very poorly, even when all words in the document are used. In contrast, both semantic methods perform very well unless a very small percentage of words is included in the query (i.e., less than 10%). Performance is even more impressive if we consider that the words included in the search query were sampled at random and, therefore, can include few or no directly relevant content words.

In summary, when the search query is composed of words from the document, the semantic and nonsemantic vector methods perform well. However, when the search query is composed of semantically related words, the semantic methods perform much better than the nonsemantic method. Of course, the



Fig. 2 Simulation 1: Target document retrieval using the BEAGLE-RP, BEAGLE, nonsemantic, and word match methods. Performance is shown as a function of the percentage of words sampled to the search query



Fig. 3 Simulation 2: Target document retrieval for queries constructed with semantic associates using the semantic and nonsemantic vector models. Performance is shown as a function of the percentage of words sampled to the search query

advantage has very strong practical importance: Users should be able to express the intent of their search without needing to use the exact words in the document they are searching for.

Simulation 3: Relationship between semantic and nonsemantic search

Simulations 1 and 2 provided initial evidence that a document can be recovered better using semantic than nonsemantic search. However, the results have been limited to expressing the difference in recovery of a single target document. To broaden our analysis, we conducted a simulation to measure the extent to which document rankings produced by a semantic search correspond to the document rankings produced by the nonsemantic and word match searches.

Simulation 3 was a repetition of Simulation 1, but we measured the agreement (i.e., Spearman rank correlation) between document ranks that were returned for all 27,560 documents using the BEAGLE-RP, BEAGLE, nonsemantic, and word match methods.

The simulation results are presented in Fig. 4. Error bars show one standard deviation above and below each mean.

There are two key results in Fig. 4. First, agreement between the retrieval profiles using all methods improved with the percentage of words included in the search query. Second, the strength of agreement was largely consistent between the different methods, with one key exception: the two semantic methods (i.e., BEAGLE-RP and BEAGLE) agreed very strongly.

We concluded that using semantic vectors is not crucial to retrieving a particular target document (see Simulation 1), but using semantic vectors retrieves a different ranked profile of related documents—even when words were not replaced by semantic associates in the search query. To the extent that the methods disagree, the semantic method returns different documents than the nonsemantic and word match methods. However, the agreement between BEAGLE-RP and BEAGLE shows that the particular instantiation of the semantic theory used to derive the underlying word meanings has only a modest influence on the documents retrieved.

Taken together, Simulations 1–3 establish that our semantic search engine can find a target document, that results are consistent with BEAGLE and BEAGLE-RP, and that semantic search differs from nonsemantic and word match searches. Admittedly, none of the results reported provide information about what documents our semantic search engine retrieves—they merely establish feasibility.

In the next section, we implement the method in a search engine interface and show that it can be used to locate related documents and articulate the structure in the document database. Because BEAGLE vectors have a lower dimensionality than BEAGLE-RP vectors, computations of similarity are more



Fig. 4 Simulation 3: Agreement of retrieved lists using the BEAGLE-RP, BEAGLE, nonsemantic, and word match methods. Performance is shown as a function of the percentage of words sampled to the search query

efficient with BEAGLE. Therefore, we use the BEAGLE rather than BEAGLE-RP vectors for the search interface.

A search interface

Thus far, we have provided a formal description of a method for semantic representation and retrieval, but we have not offered a way to use the method. To solve the problem, we developed a search interface using the Shiny package in R. The best way to use the search engine is to download and run a local copy in RStudio that we have made available in an online repository at https://osf.io/wfcmg/files/ (see the Appendix). However, an online copy of the interface can be inspected directly from https://crumplab.shinyapps.io/ SemanticLibrarian/. The online version works the same as the copy in the repository, but takes time to download through the browser and will run slower due to data exchange.

A principal purpose of an academic search engine is to find published articles that are relevant to a search query. For example, given the query "implicit learning," one might hope to find articles covering that topic.

Figure 5 presents a screen shot of our main interface (i.e., the SemanticSearch tab), searching for "perception attention memory" using the "OR" search method. Using "OR" returns a list of the documents that are most similar to any one of the individual words in the search query. If "compound search" were used, the documents returned would be those most similar to the sum of the words in the search query (i.e., $\mathbf{q} = m_{memory} + m_{attention} + m_{perception}$).

As is shown at the bottom of Fig. 5, search for "perception attention memory" returns a list of the top semantically related documents. The most similar document appears at the top of the list, the second most similar document appears second, and so on. By default, that list is 100 documents long; how-ever, the number of articles in the search list can be contracted or expanded using the *number of articles* slider.

Also shown in Fig. 5, the search results are shown as a twodimensional plot of the most similar documents; in this example, the top 500 most related articles. This geometric representation of the search results is a two-dimensional multidimensional scaling (MDS) solution based on the cosine matrix for all documents in the graph (i.e., as selected by the *number of articles* slider).

To inspect the plot, a user can hover their mouse over any point in the space. Doing so reveals its title (e.g., in the graph, Cutting's 1983 article "Four Assumptions About Invariance in Perception" is shown). Clicking on a point in the graph presents the document's abstract and related publication information on the left side of the screen, along with a weblink to the article's information in the Google scholar database.

In addition to plotting the results in a two-dimensional and searchable graph, the interface provides the user with the option to categorize the articles in the space with *k* means clustering (i.e., as implemented using the *kmeans()* clustering function in R). The number of clusters is selected by using the *number of clusters* slider. If a user asks for more than one



Fig. 5 A screen shot of the semantic search engine using the OR search function on the search term "perception attention memory." The plot shows the results for documents published between 1890 and 2016, with k means clustering applied to divide the search results into three semantic clusters

cluster, the interface colors the points in the semantic document space to indicate documents that belong to each of k semantic clusters. Clustering serves to identify groups of articles that are related to the search term in different ways.

To illustrate, Fig. 5 presents a three-cluster solution for our search of "perception attention memory." Because we used the OR search function and because we asked for three clusters, it is unsurprising that the method finds and then discriminates articles focused on topics in perception (green triangles), attention (purple circles), and memory (yellow squares).¹ Also shown, the clustering reveals a sensible outcome concerning the relationships between clusters: Documents on perception and attention border one another, documents on attention and memory border one another, but documents on perception and memory do not. That organization of topics is consistent with our professional intuition that work on attention serves as the bridge between work on perception and memory. In our experience, selecting a small number of clusters reveals broad and meaningful distinctions whereas requesting a large number of clusters serves to over differentiate the search set in ways that become increasingly difficult to understand. Because the method is based on Monte Carlo simulation, the presented k means solution can change from request to request. However, in our experience, the solutions are pretty stable and instructive.

In summary, entering a search query retrieves the most related documents (where the number of documents returned is specified by the user), with the list being presented as an ordered list as well as a corresponding and interactively searchable two-dimensional MDS plot that can be read as the "semantic neighborhood" of the search query. Clustering tools are available to help the user organize their inspection of the local neighborhoods within the global solution and to find groups of articles that relate to the search query in different ways. Because the search interface is point and click, a user does not need to directly engage the computational underbelly of the model or develop code to render the search results in a readable form.

Document neighborhoods

The interface also supports different kinds of searches. For example, a user can select the ArticleSimilarity tab to select an article title from the database and submit that article as a search query—the interface is set up so that titles autocomplete (e.g., typing "information theory" into the search box will provide a dropdown list of articles that include both of those words). Figure 6 presents an example of article search for the article "Information Theory and Immediate Recall," authored by Aborn and Rubenstein (1952).

Submitting an article as a search term produces the same style of output that free search does: an ordered list of documents and a two-dimensional MDS plot that shows the article's neighborhood (see Fig. 6). As in free search, the *number of articles* slider allows the user to specify the number of articles returned, hovering the mouse over a point in the graph reveals the article title, and clicking on a point displays its abstract and other information in the search box at the left side of the screen. Manipulating the *number of clusters* slider colors the points in the space, helping a user locate and quickly search through local neighborhoods in the global solution.

Author neighborhoods

The search engine and interface can also be used to infer and inspect the relationships between the authors of articles in the database. To use this function, a user selects the AuthorSimilarity tab and, then, submits an author name from the database as a search term—the interface is set up so that titles autocomplete (e.g., typing "John" into the search box will provide a dropdown list of authors with the name John in their publication name).

To conduct author search, each author's representation in the semantic space, a_i , is computed as the sum of all document vectors that the author has published,

$$a_i = \sum_{j=1}^{j=d} d_j,\tag{11}$$

where a_i is the representation of author *i*, *d* is the number of documents published by author *i*, and d_j is the representation of document *j* published by author *i*. Critically, BEAGLE allows us to represent authors in the same dimensionality space as words and documents. Thus, authors can not only be compared to one another, but to individual words and documents as well (more on this shortly).

Figure 7 presents an example using the author "Vokey, John R."

Issuing an author search produces an author's semantic neighborhood in the same style of output in the free search and article search queries (see Figs. 5 and 6). However, the points correspond to authors rather than articles. Once the space is drawn, the user can manipulate the output in the familiar ways. The *number of authors* slider allows the user to specify the number of authors that appear in the semantic neighborhood. Hovering the mouse over a point reveals the author's name. Manipulating the *number of clusters* slider colors the points in the space to locate groups of authors who are similar to the target author in different ways. In our experience, author search can help users discover scientists who examine ideas

¹ If we had used the "compound" search function, the articles returned would match on all three terms, thus providing a different search set and, thus, a different clustering of documents.



Fig. 6 A screen shot of the semantic search engine to find documents similar to a specified article. The plot shows results for 100 documents published between 1890 and 2016, with k means clustering applied to divide the search results into three semantic clusters

similar to those that the users themselves are interested in, but of whom they may not previously have been aware because of differences in language use.

Article/author similarity

Finally, the search interface provides an option to find authors whose work is related to a target article. To use this function, a user can select the ArticleAuthor tab and select an article from the database of article titles—the interface is set up so that titles autocomplete (e.g., typing "implicit learning" into the search box will provide a dropdown list of articles with both words in the publication title).

Figure 8 presents an example of the output for this function, using the article "Implicit Learning and Tacit Knowledge" by Arthur S. Reber (1989).

Issuing an article/author search produces a list of authors whose work is most similar to the target article. Manipulating the *number of entries* dropdown box can be used to change the number of authors displayed in the list, ordered from most to least similar.

In summary, the search interface provides a way for scientists to inspect the psychological record by free search, article search, author search, or article/author search. The search results are presented in two ways: as an ordered list and/or as a twodimensional MDS plot. The two-dimensional plot can be

Deringer

manipulated to reveal clusters of similar articles or authors and supports an intuitive way to display and inspect the results of a search. All of the results are produced using BEAGLE (i.e., a modern theory of semantic memory) as the underlying inference engine. To the extent that BEAGLE stands as a valid theory of psychological semantics, the results that our system produces stand as a psychologically valid method for document search and retrieval.

General discussion

By tradition, document retrieval systems are premised on methods developed outside of psychological investigation. The Semantic Librarian presented here takes a different philosophical approach. Rather than reverse-engineer a computational solution, we used modern psychological theories of human semantic memory to derive a base representation of word meanings and leveraged those representations to perform semantically indexed search.

To evaluate the system, we have reported simulations to show that the system can recover a target document, is tolerant to incomplete search queries, and is superior to nonsemantic and word match methods. Encouraged by those successes, we implemented the method in a user interface that supports easy interaction with the model. We rendered the search results

SemanticLibrarian SemanticSearch ArticleSimilarity	AuthorSimilarity ArticleAuthor About 📿 Github	
Search an author author Vokey, John R.	Find similar authors Author Space ·	aat.
	Whitteen Bruce W.A.	_
	International state State International state Internation and state International state <thi< td=""><td></td></thi<>	
	Top Authors	_
	Show 10 ¢ entries Search:	, ÷
	Vokey, John R.	1
	Brooks, Lee R. 0.993	37
	Jacoby, Larry L. 0.992	27
	Glenberg, Arthur M. 0.992	23
	Bower, Gordon H. 0.991	16
	Hampton, James A. 0.991	15
	Koriat, Asher 0.991	15

Fig. 7 A screen shot of the semantic search engine to find authors who are similar to a specified name. The plot shows results for 100 authors who published articles in the database; k means clustering is applied to divide the search results into three semantic clusters

SemanticLibrarian	SemanticSearch	ArticleSimilarity	AuthorSimilarity ArticleAuthor	About	🖓 Github				
Search an A	PA article		Most similar authors to the article Takes a couple seconds to load Show 25 entries Search:						
Implicit Learning A	nd Tacit Knowledge.	•	Authors			÷		Similarity 🕴	
			Reber, Arthur S.					0.9856	
			Blanchard-Fields, Fredda					0.9752	
			Buss, Ray R.					0.9752	
			Cho, Jeung Ryeul					0.9752	
			Druhan, Barry					0.9752	
			Stanley, William B.					0.9752	
			Crossley, Matthew J.					0.9741	
		Redington, Martin					0.9739		
			Willingham, Daniel B.					0.9724	
			Billman, Dorrit					0.9716	
			Dulany, Don E.					0.9715	
		Love, Bradley C.					0.9714		
		Johnstone, Theresa					0.9711		
			Sun, Ron					0.971	
			Wright, Richard L.					0.971	
		Kahan, Steven					0.9707		
			Tunney, Richard J.					0.9707	
			Mathews, Robert C.					0.9705	
			Gureckis, Todd M.					0.97	
			Perruchet, Pierre					0.9697	
			Clapper, John P.					0.9696	
			Dewey, Gerald I.					0.9694	
			Hoffman, Aaron B.					0.9694	
			Nisbett, Richard E.					0.9693	

Fig. 8 A screen shot of the semantic search engine to find authors who are similar to a specified article. The list shows the results for 25 authors who published work in the database

using both a ranked list presentation and a two-dimensional MDS display. The MDS plots make for an intuitive visual rendering of the relationships between search terms, articles, and authors and makes inspection of similar documents quick and easy.

Our approach is grounded in a validated descriptive theory of language behavior and, so, verifies that a descriptive theory of semantics can be used to construct a meaningful search engine. However, we have not yet conducted an analysis of model fit to user intent. To do so, we would need to collect data on search behavior and subjective evaluation of the search engine from domain experts (i.e., university professors and graduate students). We plan to perform a descriptive analysis of the search engine's performance in the future and empirically evaluate the extent to which the documents it returns are meaningful and appropriate.

Although we have presented our method in the context of search engine design, it can be applied to other problems. For example, the technique can be used to match manuscripts to reviewers, to develop recommender systems for books (e.g., Johns & Jamieson, 2018), and to visualize the relationships between verbal responses by participants in qualitative research designs. However, we are most excited about the prospect that our method might be useful to researchers in the domain of computational humanities (Moretti, 2005) for conducting a large-scale analysis of content and structure in any document database (e.g., Green & Feinerer, 2015; Green, Feinerer, & Burman, 2013, 2014, 2015a, 2015b; see also Green, 2016).

In fields outside of psychology, cognitive computing research has taken an engineering-style approach to the design of cognitive systems. In that tradition, researchers identify a problem, define a goal, and engineer a system to satisfy that goal. The method we have presented here illustrates a different approach to cognitive computing. We used theories of representation and retrieval developed to understand human cognition to inform the design of an artificial cognitive system for document representation and retrieval (see also Deerwester, Dumais, Landauer, Furnas, & Harshman, 1990).

We view the difference between traditional and psychologically inspired cognitive computing to be analogous to the difference between traditional and biologically inspired engineering, in which scientists leverage the lessons and study of natural cognitive systems to solve complex applied problems. For example, Tero et al. (2010) studied how slime molds (*Physarum polycephalum*) develop efficient and fault-tolerant transportation networks (see also Zhu, Kim, Hara, & Aono, 2018, for a similar analysis in relation to computing and complex problem solving). They used that knowledge to design a computational method to design and optimize human transportation networks (e.g., rail systems). Just as Tero et al. demonstrated that the experimental study of slime molds can produce insights for the design of transportation networks, we are hopeful that our analysis helps point out that basic science conducted to understand human semantic behavior can provide productive practical insights and solutions for search engine design.

Author note This research was supported by a Discovery Grant and by a CGS-M Scholarship from the Natural Sciences and Engineering Research Council of Canada, to R.K.J. and M.T.C., respectively.

Appendix

Semantic Librarian Readme

This repository at https://osf.io/wfcmg/ contains a SHINY app of the Semantic Librarian that can be run locally on a user's computer. An online version of this app is available at http:// crumplab.shinyapps.io/athena/

Steps to run locally:

- 1. Install R and RStudio.
- 2. Download the SemanticLibrarian folder to your desktop from https://osf.io/wfcmg/.
- 3. Install the Shiny package.
- 4. Package dependencies. All of the R packages used for this Shiny app are in the Packrat folder. You may be able to run this app without first installing those packages. If not, use the list in the packrat folder to install all of the necessary packages.

Running the app:

- 5. Open the .Rproj file to load this R project into RStudio.
- 6. The Shiny app files are contained in global.R, server.R, and ui.R. Opening any of those files in RStudio should allow you to view a run app button in the text editor. Press the Run app button to run the app.

The folder SemanticLibrarianBeta contains a development version with some additional features. To run that version, make sure the .RData files in the allData folder (this version) are copied into the allData folder in the beta version.

References

- Aborn, M., & Rubenstein, H. (1952). Information theory and immediate recall. *Journal of Experimental Psychology*, 44, 260–266.
- Anderson, J. R. (2013). ACT's propositional network. In Language, memory, and thought (pp. 146–181). Psychology Press.
- Bedi, G., Carrillo, F., Cecchi, G. A., Slezak, D. F., Sigman, M., Mota, N. B., . . . Corcoran, C. M. (2015). Automated analysis of free speech predicts psychosis onset in high-risk youths. *NPJ Schizophrenia*, 1, 15030. https://doi.org/10.1038/npjschz.2015.30

- Bontcheva, K., Tablan, V., & Cunningham, H. (2014). Semantic search over documents and ontologies. In N. Ferro (Ed.), Bridging between information retrieval and databases (pp. 31–53). Berlin: Springer.
- Brooks, R. (1991). New approaches to robotics. *Science*, 253, 1227–1232. Brosowsky, N., Crump, M. J. C. (2018). You should hate this movie! Detecting concealed attitudes of online persuaders. Poster presented at the Annual Meeting of the Psychonomic Society, New Orleans.
- Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, 82, 407–428. https:// doi.org/10.1037/0033-295X.82.6.407
- Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8, 240– 247.
- Cook, M. (2018). The mathematics of clinical diagnosis: Cognitivelyinspired computational psychiatry (Master's thesis). University of Manitoba, Winnipeg, MB.
- Cutting, J. E. (1983). Four assumptions about invariance in perception. Journal of Experimental Psychology: Human Perception and Performance, 9, 310–317. https://doi.org/10.1037/0096-1523.9.2.310
- de Saussure, F. (2011). Course in general linguistics (P. Meisel & H. Saussy, Eds.; W. Baskin, Trans.). New York: Columbia University Press.
- Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41, 391–407.
- Firth, J. R. (1957). A synopsis of linguistic theory, 1930–1955. In Philological Society (Great Britain) (Ed.), *Studies in linguistic* analysis. Oxford, UK: Blackwell.
- Foltz, P. W., Laham, D., & Landauer, T. K. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1, 939–944.
- Friendly, M., Franklin, P. E., Hoffman, D., & Rubin, D. C. (1982). The Toronto Word Pool: Norms for imagery, concreteness, orthographic variables, and grammatical usage for 1,080 words. *Behavior Research Methods & Instrumentation*, 14, 375–399. https://doi. org/10.3758/BF03203275
- Gilhooly, K. J., & Logie, R. H. (1980). Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12, 395–427. https://doi.org/10.3758/BF03201693
- Graesser, A. C. (2011). Learning, thinking, and emoting with discourse technologies. *American Psychologist*, 66, 746–757.
- Green, C. D. (2016). A digital future for the history of psychology? *History of Psychology*, 19, 209–219.
- Green, C. D., & Feinerer, I. (2015). The evolution of American Journal of Psychology, 1887–1903: A network investigation. American Journal of Psychology, 128, 387–401.
- Green, C. D., Feinerer, I., & Burman, J. T. (2013). Beyond the schools of psychology 1: Digital analysis of *Psychological Review*, 1894–1903. *Journal of the History of the Behavioral Sciences*, 49, 167–189.
- Green, C. D., Feinerer, I., & Burman, J. T. (2014). Beyond the schools of psychology 2: Digital analysis of *Psychological Review*, 1904–1923. *Journal of the History of the Behavioral Sciences*, 50, 249–279.
- Green, C. D., Feinerer, I., & Burman, J. T. (2015a). Searching for the structure of early American psychology: Networking *Psychological Review*, 1894–1908. *History of Psychology*, 18, 15–31.
- Green, C. D., Feinerer, I., & Burman, J. T. (2015b). Searching for the structure of early American psychology: Networking *Psychological Review*, 1909–1923. *History of Psychology*, 18, 196–204.
- Johns, B. T., & Jamieson, R. K. (2018). A large-scale analysis of variance in written language. *Cognitive Science*, 42, 1360–1374. https://doi. org/10.1111/cogs.12583
- Johns, B. T., Taler, V., Pisoni, D. B., Farlow, M. R., Hake, A. M., Kareken, D. A., Unverzagt, F. R., & Jones, M. N. (2018).

Cognitive modeling as an interface between brain and behavior: Measuring the semantic decline in mild cognitive impairment. *Canadian Journal of Experimental Psychology*, 72, 117–126. https://doi.org/10.1037/cep0000132

- Jones, M. N., Kintsch, W., & Mewhort, D. J. K. (2006). Highdimensional semantic space accounts of priming. *Journal of Memory and Language*, 55, 534–552. https://doi.org/10.1016/j. jml.2006.07.003
- Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114, 1–37. https://doi.org/10.1037/0033-295X.114.1.1
- Kanerva, P. (1994). The spatter code for encoding concepts at many levels. In *International Conference on Artificial Neural Networks* (pp. 226–229). London: Springer.
- Kwantes, P. J., Derbentseva, N., Lam, Q., Vartanian, O., & Marmurek, H. H. (2016). Assessing the Big Five personality traits with latent semantic analysis. *Personality and Individual Differences*, 102, 229– 233.
- Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104, 211–240. https://doi.org/10.1037/0033-295X.104.2.211
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- Moretti, F. (2005). Graphs, maps, trees: Abstract models for literary history. New York: Verso.
- Murdock, B. B. (1995). Developing TODAM: Three models for serialorder information. *Memory & Cognition*, 23, 631–645. https://doi. org/10.3758/BF03197264
- Murdock, B. B. (1997). Context and mediators in a theory of distributed associative memory (TODAM2). *Psychological Review*, 104, 839– 862. https://doi.org/10.1037/0033-295X.104.4.839
- Murdock, B. B., Jr. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89, 609–626. https://doi.org/10.1037/0033-295X.89.6.609
- Murdock, B. B., Jr. (1983). A distributed memory model for serial-order information. *Psychological Review*, 90, 316–338.
- Osgood, C. E. (1952). The nature and measurement of meaning. *Psychological Review*, 49, 197–237. https://doi.org/10.1037/ h0055737
- Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6, 623–641.
- Reber, A. S. (1989). Implicit learning and tacit knowledge. Journal of Experimental Psychology: General, 118, 219–235. https://doi.org/ 10.1037/0096-3445.118.3.219
- Recchia, G., Sahlgren, M., Kanerva, P., & Jones, M. N. (2015). Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation. *Computational Intelligence and Neuroscience*, 2015, 986574. https://doi.org/10.1155/2015/986574
- Roscoe, R. D., Allen, L. K., Cai, Z., Weston, J. L., Crossley, S. A., & McNamara, D. S. (2014). The writing pal intelligent tutoring system: Usability testing and development. *Computers and Composition*, 34, 39–59.
- Roscoe, R. D, & McNamara, D. S. (2013). Writing pal: Feasibility of an intelligent writing strategy tutor in the high school classroom. *Journal of Educational Psychology 105*, 1010–1025.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386–408. https://doi.org/10.1037/h0042519
- Rubin, D. C., & Friendly, M. (1986). Predicting which words get recalled: Measures of free recall, availability, goodness, emotionality, and pronounceability for 925 nouns. *Memory & Cognition*, 14, 79–94.

- Rubin, T., Koyejo, O., Jones, M. N., & Yarkoni, T. (2016). Generalized correspondence-LDA models (GC-LDA) for identifying functional regions in the brain. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), Advances in neural information processing systems (pp. 1126–1134). Red Hook: Curran Associates.
- Rubin, T. N., Koyejo, O., Gorgolewski, K. J., Jones, M. N., Poldrack, R. A., & Yarkoni, T. (2017). Decoding brain activity using a large-scale probabilistic functional–anatomical atlas of human cognition. *PLoS Computational Biology*, 13, e1005649. https://doi.org/10.1371/ journal.pcbi.1005649
- Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 45–76). Cambridge, MA: MIT Press.
- Sahlgren, M., Holst, A., & Kanerva, P. (2008). Permutations as a means to encode order in word space. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), Proceedings of the 30th Annual Conference of the

Cognitive Science Society (pp. 1300–1305). Austin: Cognitive Science Society.

- Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D. P., Fricker, M. D., ... Nakagaki, T. (2010). Rules for biologically inspired adaptive network design. *Science*, 327, 439–442.
- Toglia, M. P., & Battig, W. F. (1978). Handbook of semantic word norms. Hillsdale: Erlbaum.
- Zhu, L., Kim, S.-J., Hara, M., & Aono, M. (2018). Remarkable problemsolving ability of unicellular amoeboid organism and its mechanism. *Royal Society Open Science*, 5, 180396.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.