Lexical Semantics with Vector Symbolic Architectures

Adam Roussel
Ruhr University Bochum
Department of Linguistics
roussel@linguistics.rub.de

Abstract

Conventional approaches to the construction of word vectors typically require very large amounts of unstructured text and powerful computing hardware, and the vectors themselves are also difficult if not impossible to inspect or interpret on their own. In this paper, we introduce a method for building word vectors using the framework of vector symbolic architectures in order to encode the semantic information in wordnets, such as the Open English WordNet or the Open Multilingual Wordnet. Such vectors perform surprisingly well on common word similarity benchmarks, and yet they are transparent, interpretable, and the information contained within them has a clear provenance.

1 Introduction

Semantic representations based on word vectors are the foundation of many, if not most, algorithms in computational linguistics that require some means of handling lexical semantics. However, the most widespread of these, word embeddings based on the prediction of masked words, have a number of significant limitations.

These begin with how such embeddings are constructed. The training process for these vectors requires large amounts of textual data, often on the order of billions of tokens. The scale of data required then brings further problems along with it: It is difficult (often prohibitively) to audit and improve the training data, the large size of the datasets requires accordingly high performance computers capable of processing the data, and models can only be trained for languages and domains where such large quantities of naturally occurring data exist.

Then, once a model has been trained, the resulting vectors are difficult if not impossible to fully understand and interpret. Individual vectors cannot be inspected by themselves, they can only be compared with one another in terms of their distance from one another, and it is unclear exactly what relation this distance represents: Is it more like similarity or relatedness or something else?

In this paper, we propose a way of building word vectors that aims to address these issues.[1] Our word vectors use a single manually constructed linguistic resource (though there is no reason why one should be limited to a single resource, in theory) and thus represent relationships between words whose provenance is clearly traceable. As the resource is updated and improved, or adapted to a particular domain, the vectors can be adjusted accordingly, even individually and incrementally, incorporating the new information.

By constructing the vectors in this way, rather than from unstructured text, we can avoid some of the problems described above. The vectors described in this paper can be constructed on ordinary laptop computers and require no more than a few hundred megabytes of memory. Since they are based on linguistic resources, it is much easier to know what information goes into the vectors and how it got there. For some languages, especially ancient and historical languages, it's much more feasible to extend a linguistic resource than to discover more text from which to build a corpus.

The vectors we describe in this paper are built using techniques known as *vector symbolic architectures* or *hyper-dimensional computing*, which we apply to an established high-quality linguistic resource, the Open English WordNet (OEWN) (McCrae et al., 2019). We'll describe how the vectors are constructed (Section 2.2) and some of their interesting properties relating to interpretability (4.1). Then we'll compare their performance on benchmarks for word embeddings to get an idea

[1]The code for the experiments described here can be found at `https://git.noc.rub.de/ajroussel/vsa-wn`.

Symbol	Vector	Sim. to x
x	00110000...	1.0
y	01001101...	−0.022
z	00010011...	0.001
$x + y + z$	00010001...	0.496
$z \otimes x$	00100011...	0.001
$z \oslash (z \otimes x)$	00110000...	1.0

Table 1: Examples of some of the basic VSA operations.

of their ability to be used in similar applications to word embeddings (4.2) and assess how well this approach generalizes to other wordnets (4.3). Finally, we conclude with some discussion of the experimental results.

2 Methods

2.1 Vector Symbolic Architectures

Vector symbolic architectures (VSA), also referred to as *hyper-dimensional computing* (Kanerva, 2009; Neubert et al., 2019; Schlegel et al., 2022; Kleyko et al., 2022), constitute an approach to computing that is characterized by the use of very high-dimensional random vectors (usually several thousands of dimensions), which are taken to represent symbols, since such large random vectors are almost certain to be nearly orthogonal and thus easily distinguishable.

These symbolic vectors are then combined using particular operations enabling computation more generally. The most important of these operations are *bind*, *bundle*, and *protect* (also termed *binding*, *superposition*, and *permutation*, e.g. Kleyko et al. (2022)). Binding results in a new vector that is dissimilar to both input vectors, and bundling combines two or more vectors to result in a new vector that is similar to all of the input vectors. Protecting is a unary roll operation that shifts all values one or more places further in the vector and results in a vector that is nearly orthogonal to the input vector. Binding can be used for the encoding of vectors into key-value pairs, from which values are recoverable by means of an inverse operation, *unbind* (represented here by $\oslash$), and bundling can be used to combine vectors into set-like data structures.

Though there are many possible instantiations of these parameters, some using integers, real numbers, or complex numbers, for our model, we use binary vectors of 8192 (or 2^{13}) dimensions. Such vectors can be efficiently packed, with eight dimensions per byte, so that they use relatively little memory. In this particular VSA variant, termed *binary spatter code* (Schlegel et al. (2022) and originally described in Kanerva (1996)), the bind operation ($\otimes$) is bitwise XOR, bundle is the majority rule applied elementwise (with ties broken randomly), and the similarity metric is the inverse Hamming distance (normalized to $[-1, 1]$). In this variant, the inverse of the bind operation ("unbind", $\oslash$) is the same, also XOR. Table 1 summarizes these basic properties.

Vector symbolic architectures have many interesting properties, and they could be helpful in bringing some of the advantages of statistical approaches to symbolic computing. Since the information stored in the high dimensional vectors employed by VSAs is highly redundant, the vectors are very robust to noise and ambiguity: Many bits could be lost before the system's behavior starts to degrade (cf. Section 4.4). Particularly important for linguistic applications, VSAs also promise to enable graded similarity judgments in otherwise symbolic algorithms, lending approaches that might otherwise be too brittle some flexibility.

2.2 Building a lexical semantic model

The source of the information stored in the model is a single WordNet-like lexical resource, such as the Open English WordNet 2021 (McCrae et al., 2019), which we will use here for the first set of experiments. The model works on the assumption that the most important synsets for determining the meaning of some synset x are those closest to it, from 1 to some small number n steps away. The closer these other synsets are, the more relevant they are to the meaning of x. Very close synsets will share similar neighborhoods in the network – i.e., they participate in many of the same relations with the same target synsets – and will thus have a high similarity to one another.

Each relation and each synset is first assigned a new random vector, an "elemental" vector. Then the corresponding "semantic" vectors are constructed by traversing the network (a digraph), breadth-first, out from each synset in turn, up to some maximum depth n. We will use E to denote the function from a given synset or relation to its elemental vector and S for the function from a synset to its semantic vector. In traversing the

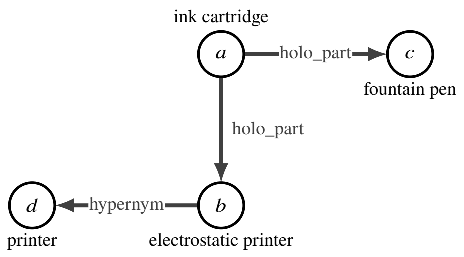

Figure 1: Example of a synset with its immediate neighbors.

network, we include all of the relation types that are present, but only in one direction. Many of the relations have a canonical inverse, so for the relation pairs hyponym–hypernym and mero_part–holo_part, for example, we only include hypernym and holo_part.

As the distance from x increases, the strength of a relation can be carefully modulated via the introduction of controlled amounts of random noise. This is the purpose of the attenuation function $w(\beta, v_i, d)$ given in (1) below. This function will, for each element v_i of some vector v, invert the value of that element at random with a probability derived from β. As the distance d from a given node increases, the more distant target nodes are more weakly associated with that relation, since there is a higher probability that a bit in the relation vector will be flipped.

$$w(\beta, v_i, d) = \begin{cases} 1 - v_i & \text{if } \mathcal{U}(0,1) > \beta^d \\ v_i & \text{otherwise} \end{cases} \tag{1}$$

Applying this attenuation function, the semantic vectors are constructed as in (2):

$$S(x) = \sum_{i=1}^{n} \sum_{r \in R} w(\beta, E(r), i-1) \otimes E(y) \tag{2}$$

Where R is the set of edges (i.e. relations) at a specific distance i from x. At each step away from x, we collect the edges that are present and the target nodes y that they point to and bind these together, attenuating the corresponding relation vector $E(r)$ according to the distance from x to y and stopping when the maximum depth n has been reached. The semantic vector $S(x)$ contains all of these components bundled together in superposition. The full model then consists of three clean-up memory instances, containing the original relation, elemental, and semantic vectors, which are used to translate vector representations back to relation names or synsets as needed.

For the example in Figure 1, the semantic vector for a, the synset for *ink cartridge*, would be constructed as follows:

$$\begin{aligned} S(a) &= E(\text{holo\_part}) \otimes E(b) \\ &+ E(\text{holo\_part}) \otimes E(c) \\ &+ w(0.95, E(\text{hypernym}), 1) \otimes E(d) \end{aligned} \tag{3}$$

3 Related work

3.1 Non-distributional word vectors

Other approaches to creating word vectors based on non-distributional lexical information include Faruqui and Dyer (2015), Saedi et al. (2018), and Kutuzov et al. (2019).

In Faruqui and Dyer (2015) the authors use a relatively large collection of linguistic resources to compile semantic representations for English words. This includes WordNet, but also FrameNet, sentiment polarity and word connotation databases, word-color association lexicons, treebanks, and Roget's thesaurus. Each vector has a dimension corresponding to a particular feature value for all of features derived from these resources. The resulting 172,418-dimensional vectors can then be reduced using singular value decomposition, however the large and very sparse vectors are nevertheless computationally easy to work with and have the advantage of greater interpretability.

The use of such a variety of rich linguistic resources is at once a strength and shortcoming of this approach: The vectors contain lots of useful information for various tasks, however it is dependent on all of these different resources, which makes it considerably more difficult to transfer the approach to new languages, since it is unclear how many of the resources would need to be present in order to have usable vectors.

In contrast, Saedi et al. (2018) construct their embeddings solely based on the relational structure of WordNet, similarly to the approach described in this paper. They begin with an $N \times N$ adjacency matrix, containing the value 1 for links between immediately connected nodes. Then to these values they add links of length 2, which are reduced using a decay factor α^n with $n = 2$ (this is similar to our use of the attenuation function described in (1) above). As the length of the paths increases and the decay factor approaches zero, the values in the adjacency matrix would eventually converge. The result of this convergence is calculated analytically, involving an inverse matrix operation. Finally, the word vectors in this matrix are reduced to 850 dimensions by using principal components analysis to transform the matrix.

As the calculation of the word embeddings involves computationally costly matrix inversion operations, it was infeasible to build a model for all roughly 120,000 synsets contained in WordNet, such that the authors were limited to including only 60k of the synsets. Even then, inverting the 60k-dimensional matrix used all of the 32 CPUs and 430 GB of RAM they had available (it is unknown for how long). The authors experiment with a range of subgraphs, including random subgraphs of 25k–60k synsets as well as a subgraph of the 13k most frequent synsets. These much smaller subgraphs nevertheless result in word embeddings that perform only slightly worse or just as well on SimLex-999.

Kutuzov et al. (2019) employ a machine learning approach called *path2vec* to deriving a set of node embeddings from a graph, such as WordNet. The basic idea is that their model will learn vectors such that the similarity between the vectors approximates the similarities given by some user-selected graph similarity metric, such as path similarity or Leacock–Chodorow (LCH) similarities. The learning approach is similar to the skipgram model of word2vec, but instead of optimizing positive pairs towards 1 and negative (random) pairs towards 0, all pairs are drawn from the graph to be embedded and optimized towards the values of the given similarity metric. They only train and evaluate the model for noun synsets, since these are covered better in WordNet than the other parts of speech, but on the 666 noun pairs in SimLex-999, their *path2vec* embeddings outperform the plain path similarity measure, reaching a correlation of 0.555 with the human ratings.

Their motivation for deriving vectors for the nodes of a graph is that actually computing most node similarity measures involves the computationally costly and inefficient traversal of the graph, but it is much faster to compare two vectors. The authors report that, to calculate the 82,115 similarity values between one noun and all of the other nouns, it took 30 s using the LCH measure but just 0.007 s to perform the same number of comparisons using float vectors like those produced by *path2vec*.

Though the machine learning methods employed in Kutuzov et al. (2019) are very different from the VSA approach we describe here, comparing the VSA representations of synsets is similarly much faster than computing path similarities directly. However, as the authors also report, computing Hamming distance between binary vectors (as in the VSA variant we employ) is less efficient than computing the dot product between float vectors.

3.2 Semantic representations using VSAs

The model described here is constructed using a technique that is similar in many respects to *predication-based semantic indexing*, as described by Widdows and Cohen (2015). The main difference is in how the semantic vectors are constructed: There, only relations that directly involve x are included in $S(x)$, whereas our model includes more distant relations, with greater attenuation of the more distant relations – which seems necessary to capture the relationships between synsets as opposed to the semantically richer relations contained in the biomedical database described there.

Cohen et al. (2013); Widdows and Cohen (2015) also describe the purpose and use of “demarcator” vectors, which are designed to be new random vectors that have a selectable degree of similarity (“measured distance”) to certain other vectors. These were the inspiration of the attenuation function used here. Whereas the purpose of demarcator vectors was to position character vectors at regular intervals between two endpoints, we use a similar technique to gradually decay relation vectors for more distantly related word senses.

4 Evaluation

4.1 Querying the model

Lexical semantic models constructed according to this method can be inspected and queried by the application of particular operations.

If the vector for the hypernym relation $E(\text{hypernym})$ is unbound from a semantic vector (note that for binary VSAs, this is the same operation as binding), the resulting vector will be nearest to the vector that was originally bound to that semantic vector:

$$S(\text{hammer}) \oslash E(\text{hypernym}) \approx E(y) \qquad (4)$$

Where y stands for the synset to which *hammer* has the hypernym relation. These are the nearest neighbors to the query vector in (4):

0.634	hand tool
0.314	tool
0.305	implement

This accurately reflects that the best fit for "hypernym of hammer" is *hand tool* (this is a direct relation between the two synsets in the OEWN), and it also shows how there are other alternative answers to the query: hypernyms higher up beyond *hand tool*.

In a similar fashion, queries can be constructed to determine the relation between two vectors. To construct the necessary query vector, we just reverse the operation that was used to construct some semantic representation.

$$S(\text{hammer}) = E(\text{hypernym}) \otimes E(\text{hand tool}) + \dots \quad (5)$$

$$S(\text{hammer}) \oslash E(\text{hand tool}) \approx E(\text{hypernym}) \quad (6)$$

By this principle, analogies can also be modelled mathematically with only a handful of vectors, as shown in Kanerva (2010). Similar to the query vectors above, the idea is to construct a vector which is similar to the one you seek by binding or unbinding the necessary components. For our model, as with the PSI model in Widdows and Cohen (2015), we have to account for the additional distinction between semantic and elemental vectors, namely that the relational information is contained in semantic vectors and "points" to elemental vectors, so the analogy queries must be constructed slightly differently, but the principle is the same.

By unbinding $E(y)$ from some vector $S(x)$, we can expect a vector that is similar to the relations that hold from x to y. If this relation-like vector is applied to a new target $E(y')$, then the result is similar to some $S(x')$ that bears this relation to the new target. Thus, to complete the analogy, "apple tree is to apple as pear tree is to X", we can use the following query vector:

$$S(\text{apple}) \oslash E(\text{apple tree}) \otimes E(\text{pear tree}) \quad (7)$$

The nearest vector to this query vector is the one for *pear*, $S(\text{pear})$, and, in fact, this method retrieves a number of different kinds of pear:

0.145	pear
0.113	bosc
0.113	anjou
0.113	bartlett pear
0.113	seckel pear
0.113	Clapp's Favourite

Upon closer inspection, we see that the relation that this query seems to be picking up on, holo_part, is only present in the synsets for apple and pear, and the relation vector is not in the query vector explicitly, rather it is calculated by (7). Interestingly, the relation is also not present in the other nearest neighbor synsets, which are found due to their similarity to *pear* (and this similarity is due to their being hyponyms of *pear*).

4.2 Comparison with benchmarks

In order to get an idea of the quality and potential utility of the vectors constructed according to the method described here, we tested our model against a set of established benchmarks that are commonly used to evaluate word meaning representations. These include the SimLex-999 dataset (Hill et al., 2015), which focuses on similarity (and not relatedness) and the MEN dataset (Bruni et al., 2012), focusing more on semantic relatedness. Then we use both the similarity and relatedness sections of the WordSim353 datasets from Agirre et al. (2009). The datasets consist of word pairs that have been assigned either similarity or relatedness ratings by human judges (inter-rater correlation on the SimLex-999 dataset was in the 0.67–0.78 range, depending on how measured), using differing scales. All of the comparisons given in this paper use Spearman's ρ.

Heuristics One practical difficulty in evaluation is that relations in wordnets hold between synsets, and the VSA model we wish to test is therefore also synset-oriented, yet all of these datasets concern words or lemmas. It is impossible to know after the fact which meaning or meanings the raters had in mind when they were recording their ratings, so it is actually impossible in principle to choose the correct word sense and thus the appropriate synset vector. Therefore, for the purposes of this evaluation, we tested and compared a few heuristics for choosing or making a vector for a given lemma from a list of synset vectors.

The "first" heuristic is to compare the first synset listed for each lemma, which operates on the assumption that the synsets are listed in a particu-

	Similarity		Relatedness	
Model	SimLex-999	WS353-S	MEN	WS353-R
word2vec	0.44	0.74	0.70	0.61
Saedi et al. (2018)	0.50	0.65	0.46	0.32
Path similarity	0.314 (0.468)	0.549 (0.610)	0.283 (0.311)	−0.003 (−0.039)
VSA, $d = 3, n = 8192$	0.421	0.455 (0.495)	0.271 (0.287)	0.137
VSA, $d = 5, n = 8192$	0.442	0.590	0.381	0.156
VSA, $d = 7, n = 8192$	0.384 (0.417)	0.538 (0.565)	0.385 (0.386)	0.152
VSA, $d = 9, n = 8192$	0.344 (0.400)	0.514 (0.550)	0.371 (0.378)	0.131

Table 2: Performance of various model iterations and baselines on benchmark datasets, in terms of Spearman's ρ. Values above the line are from Saedi et al. (2018). Values under the line use the "average" sense selection heuristic, and when the value using the "max" heuristic is greater, it is given in parenthesis.

lar order, perhaps according to corpus frequency, though it is unclear to what extent this is the case or what corpus might have been involved. The second, "average", is simply to compare all of the synsets of the first lemma with all of the synsets of the second lemma and then to take the average of the resulting similarity scores. Finally, there's the "max" heuristic, in which all pairs are compared as in the "average" heuristic, except that we then take the maximum similarity score instead of the average. Here the intuition is that the raters would have subconsciously chosen the most relevant sense of, say, *bank* when the comparison was with *money*.

Discussion We compare our results with three other types of scores: There are the scores of the system system described in Saedi et al. (2018) as well as the word2vec (Mikolov et al., 2013) scores with which the authors of that paper compared their system. Then we compare our results with a simple baseline: The path similarity score, which approaches 0 as the shortest path between two synsets increases in length. The path similarity has its limitations: The two nodes in question must be connected (this can be compensated by the addition of a fake root node connecting the other root nodes), and they must be of the same part of speech. This last requirement is mostly an issue for the relatedness benchmarks, which contain numerous pairs of differing POS.

For each system variant and baseline, we calculated a similarity score using each of the three heuristics. The "first" heuristic was consistently significantly worse than the other two heuristics, so we omit it from the comparisons here. The scores using the "average" and "max" heuristics were often fairly close, with each being sometimes greater than the other. In Table 2, we list scores using the "average" heuristic, which seemed to be more stable overall and which worked best with the system configuration that seemed to strike the best balance between the different datasets ($d = 5$). But, where the "max" heuristic performed better, it is included in parentheses.

Considering the overall results in Table 2, the VSA-based representation reflects a fair amount of correlation with the ratings in the benchmark datasets and similar levels of correlation with the comparison systems and baselines, which suggests that the vectors do capture a degree of useful semantic information in a general sense. It seems plausible that these vectors could be treated similarly to word embeddings in certain applications. Most of the time, the correlations for the system introduced here are within 0.10 of the baseline or the other systems. Particularly, on the MEN dataset, which contains relatedness scores for 3000 lemma pairs, we see significant improvement over the path similarity baseline. This is probably partly attributable to the limitations of the path similarity metric, which requires a path between the nodes to exist and the words to have the same POS, however it can also be seen as an advantage of using a vector representation for this task, despite the vectors ultimately being based on the same underlying information.

For these comparisons, we varied the maximum depth d that is used to traverse the graph when collecting relations that are to be factored in to a given semantic vector. It appears that the variants with a lower maximum depth and thus fewer distant relations performed better on the similarity benchmarks. A higher maximum depth seems beneficial on the MEN dataset, which includes ratings of relatedness, so the inclusion of more distant relations makes intuitive sense. A maximum depth towards the middle of the range, $d = 5$ seems to provide good overall performance.

4.3 Effects of wordnet size

Of course, the OEWN is not typical in terms of its scale. In order determine how well this method might work for other wordnets and other languages, we used the wordnets included in the Open Multilingual Wordnet (OMW) (Bond and Foster, 2013) project to construct several new models for various languages and evaluated these by comparing them against the Multi-SimLex benchmark dataset (Vulić et al., 2020). The Multi-SimLex dataset builds upon SimLex-999, increasing the number of word pairs to 1888 and establishing a translation and rating methodology that is specifically designed to be consistently transferable to additional languages. There are currently six languages that are covered by both Multi-SimLex and OMW; these are the ones included in the results in Table 3.

The wordnets for these six languages vary in size from large wordnets, such as FinnWordNet, which is roughly as large as the OEWN, to mid-sized wordnets, such as WOLF (Wordnet Libre du Français), to small wordnets, such as the Hebrew Wordnet. As one would expect, the larger and more complete wordnets tend to result in vectors that correlate better with the human judgments.

The results suggest that a useful degree of correlation can be expected once a wordnet contains about 50,000 synsets. However, since most of the wordnets in OMW (expand-style wordnets) benefit from a common underlying set of lexical relations between the synsets, it is probably more likely that the most important factor for the numbers in Table 3 is simply whether or not the given words are present in the wordnet at all. Since most of the word pairs are not found in the smaller wordnets, it is unsurprising that so little correlation can be observed. But what this would also mean is that adapting an existing wordnet or creating a new one need not necessarily have 50,000 or more synsets, so long as the important lexical items are covered for a given application.

4.4 Vector size

The storage capacity of n-dimensional binary vectors is theoretically quite large, with 2^n distinct possible patterns (Neubert et al., 2019). Experiments have shown that vectors with approximately 500 dimensions can bundle around 50 distinguishable vectors (Schlegel et al., 2022).

Looking at the results in Table 4, it would appear that the vectors could have been quite a lot smaller than 8192 dimensions, since the correlations with the benchmark datasets remain more or less the same, even when the vectors are much smaller. Only once the vector size has been reduced to 512 do we consistently see a noticeable reduction in the correlations. Though it is surprising that just 512 bits seem to be able to store enough information to perform this well on the benchmarks, if we were to inspect the vectors, some issues come to light.

The analogy example shown above in Section 4.1, which works well with $n = 8192$, still works similarly well with just 4096 dimensions. With 2048, there are fewer kinds of pear retrieved, but the closest answer is the same as before. But starting at 1024 dimensions, the cracks start to appear: Right after a few types of pear, things like *mock* (oewn-01227189-n) or *sidereal* (oewn-02808231-a) start to appear, with no discernible drop-off in similarity scores to indicate the boundary between what can be construed as an answer and noise. Finally, at 512 dimensions, this analogy no longer seems to work. The closest synsets there relate to types of *twayblade* (oewn-12091760-n), followed closely by a variety of seemingly random concepts, as it seems the noise has taken over.

4.5 Performance characteristics

All of the models described in this paper can each be built in a matter of minutes on an ordinary consumer-grade laptop. The models using larger 8192-dimensional vectors occupy a little over 200 MB on disk or a little over 300 MB once loaded into memory. Depending on the application, one may be able to use smaller vectors, which take up much less space.

5 Conclusion and Outlook

In this paper, we described how techniques of VSAs could be applied to linguistic resources, such as wordnets, in order to create word vectors that can be used in a manner akin to word embeddings, but which offer some key advantages over conventional word embeddings.

The performance of these vectors on established word similarity benchmarks is similar to other WordNet-based methods, if not strictly equal

Language	eng	fin	fra	cmn	pol	ara	heb
Wordnet	oewn	omw-fi	omw-fr	omw-cmn	omw-pl	omw-arb	omw-he
No. of synsets	120039	116763	59091	42312	33826	9916	5448
OOV	28	180	194	860	939	1269	1697
Path similarity	0.482	0.416	0.350	0.154	0.065	0.138	0.034
VSA, $d = 5, n = 8192$	0.434	0.444	0.388	0.153	0.126	0.196	0.070

Table 3: Correlation with human ratings in the Multi-SimLex dataset. Languages are identified by their ISO 639-3 language code, and wordnets by the specifier used by the Python `wn` package. The baseline path similarities were calculated using the "max" heuristic, and the VSA similarities using the "average" heuristic.

n	SimLex	WS353-S	MEN	WS353-R
8192	0.442	0.590	0.381	0.156
4096	0.438	0.548	0.383	0.140
2048	0.421	0.576	0.371	0.174
1024	0.429	0.537	0.344	0.125
512	0.427	0.512	0.334	0.111

Table 4: Correlation with benchmarks as vector size is varied. All models use a maximum depth of $d = 5$ and the "average" heuristic.

to them, but, in exchange for some 10 percentage points of correlation, the vectors produced by the method described here are inspectable, interpretable and can be improved incrementally. Furthermore, in contrast to the more conventional approaches, these vectors require few resources and can be built using ordinary computer hardware.

In principle, such models can be constructed from any wordnet. Though larger wordnets such as the OEWN contain many synsets and cover more lexical items, it appears that even wordnets with much fewer synsets can still leverage the underlying structure of the OMW such that the derived vector representations can be useful. This suggests that the approach described here could be valuable in situations in which it is infeasible to compile a large enough corpus for usable word embeddings or in which one seeks more transparency as to the contents of word vectors than distributional methods generally allow.

Future work Since the collection of semantic vectors functions somewhat like a queryable database, it would be interesting to investigate to what degree query vectors and comparison operations could be constructed in order to try and query relations other than similarity: For instance, by examining a synset's distance from nodes that function as paradigm words for concreteness, similar to approaches described by Turney et al. (2011), whether one could use this same model to compare the concreteness of concepts. Alternatively, perhaps it could be possible to compare vectors that have been modified in some systematic way, perhaps certain relations have been unbound from the vectors, in order to emphasize some particular aspect of their semantics or to retrieve some particular relation between them.

Sometimes there is little to distinguish some synsets from one another, since the set of relations they participate in is very similar or the same, so that our current model sees them as equivalent. This is apparent in the query (7), where all of the second-best answers are presented as equally close, which is a little counterintuitive. One idea would involve making use of the definition text as a useful workaround. For example, one could observe which words occur in which definitions and infer new relations for definitions that share significant overlap.

Finally, there is a lot of potential in encoding the semantic information in wordnets with VSAs in ways that are very different from the one we implemented in this study. Exploring some of these alternative approaches is a promising avenue for future work.

Acknowledgments

Thanks to the reviewers for their detailed and helpful feedback. Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – SFB 1475 – Projektnummer 441126958.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado. Association for Computational Linguistics.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Sofia, Bulgaria. Association for Computational Linguistics.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea. Association for Computational Linguistics.

Trevor Cohen, Dominic Widdows, Manuel Wahle, and Roger Schvaneveldt. 2013. Orthogonality and orthography: Introducing measured distance into semantic space. In *Proceedings of the Seventh International Symposium on Quantum Interaction*, pages 34–46, Leicester, UK.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 464–469, Beijing, China. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Pentti Kanerva. 1996. Binary spatter-coding of ordered k-tuples. In *Proceedings of the 1996 International Conference on Artificial Neural Networks (ICANN 96)*, pages 869–873.

Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1:139–159.

Pentti Kanerva. 2010. What we mean when we say "What's the dollar of Mexico?": Prototypes and mapping in concept space. In *Quantum Informatics for Cognitive, Social, and Semantic Processes, Papers from the 2010 AAAI Fall Symposium*, volume FS-10-08 of *AAAI Technical Report*, Arlington, VA, USA. AAAI.

Denis Kleyko, Mike Davies, Edward Paxon Frady, Pentti Kanerva, Spencer J. Kent, Bruno A. Olshausen, Evgeny Osipov, Jan M. Rabaey, Dmitri A. Rachkovskij, Abbas Rahimi, and Friedrich T. Sommer. 2022. Vector symbolic architectures as a computing framework for emerging hardware. 110(10):1538–1571.

Andrey Kutuzov, Mohammad Dorgham, Oleksiy Oliynyk, Chris Biemann, and Alexander Panchenko. 2019. Learning graph embeddings from WordNet-based similarity measures. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 125–135, Minneapolis, Minnesota. Association for Computational Linguistics.

John P. McCrae, Alexandre Rademaker, Francis Bond, Ewa Rudnicka, and Christiane Fellbaum. 2019. English WordNet 2019 – An open-source WordNet for English. In *Proceedings of the 10th Global WordNet Conference (GWC 2019)*, pages 245–252, Wrocław, Poland.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Peer Neubert, Stefan Schubert, and Peter Protzel. 2019. An introduction to hyperdimensional computing for robotics. *Künstliche Intelligenz*, 33:319–330.

Chakaveh Saedi, António Branco, João António Rodrigues, and João Silva. 2018. WordNet embeddings. In *Proceedings of the Third Workshop on Representation Learning for NLP*, pages 122–131, Melbourne, Australia. Association for Computational Linguistics.

Kenny Schlegel, Peer Neubert, and Peter Protzel. 2022. A comparison of vector symbolic architectures. *Artificial Intelligence Review*, 55:4523–4555.

Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 680–690, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ivan Vulić, Simon Baker, Edoardo Maria Ponti, Ulla Petti, Ira Leviant, Kelly Wing, Olga Majewska, Eden Bar, Matt Malone, Thierry Poibeau, Roi Reichart, and Anna Korhonen. 2020. Multi-SimLex: A large-scale evaluation of multilingual and crosslingual lexical semantic similarity. *Computational Linguistics*, 46(4):847–897.

Dominic Widdows and Trevor Cohen. 2015. Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL*, 23(2):141–173.