Hyperdimensional Utterance Spaces

A more transparent representation for situated language understanding

Jussi Karlgren
Gavagai & KTH Royal Institute of Technology, Stockholm, Sweden
jussi@kth.se

Pentti Kanerva
Redwood Center for Theoretical Neuroscience, UC Berkeley
pkanerva@csli.stanford.edu

ABSTRACT

Human language has a large and varying number of features, both lexical items and constructions, which interact to represent various aspects of communicative information. High-dimensional semantic spaces have proven useful and effective for aggregating and processing lexical information for many language processing tasks. This paper describes a hyperdimensional processing model for language data, a straightforward extension of models previously used for words to handling utterance or text level information. A hyperdimensional model is able to represent a broad range of linguistic and extra-linguistic features in a common integral framework which is suitable as a bridge between symbolic and continuous representations, as an encoding scheme for symbolic information and as a basis for feature space exploration. This paper provides an overview of the framework and an example of how it is used in a pilot experiment.

CCS CONCEPTS

• **Information systems** → **Content analysis and feature selection**; • **Computing methodologies** → **Knowledge representation and reasoning**; *Natural language processing*;

KEYWORDS

hyperdimensional computing, utterance-level semantics, constructional grammar, knowledge representation and reasoning

1 REPRESENTING LANGUAGE

Language is a general-purpose representation of human knowledge, and models to process it vary in the degree they are bound to some task or some specific usage. Currently, the trend is to learn regularities and representations with as little explicit knowledge-based linguistic processing as possible, and recent advances in such general models for end-to-end learning to address linguistics tasks have been quite successful. Most of those approaches make little use of information beyond the occurrence or co-occurrence of words in the linguistic signal and take the single word to be the atomary unit.

Jussi Karlgren's work was done as a visiting scholar at the Department of Linguistics at Stanford University, supported by a generous VINNMER Marie Curie grant from VINNOVA, the Swedish Governmental Agency for Innovation Systems.

DESIRES 2018, August 2018, Bertinoro, Italy

This proposed model for representing linguistic information will allow the handy and transparent representation of linguistic items beyond the single word and extra-linguistic data to enrich the linguistic signal in a computationally convenient representation similar to what is currently the standard processing model for word-level information.

1.1 Requirements for a representation

There are some basic qualities we want a representation to hold to. A representation should have descriptive and explanatory power, be practical and convenient for further application, be reasonably true to human performance, provide defaults to smooth over situations where a language processing component lacks knowledge or data, and provide constraints where the decision space is too broad.

Neurophysiological plausibility We want the model to be *non-compiling*, i.e. not need a separate step to accommodate a new batch of data. We want it to exhibit bounded growth, not to grow too rapidly with new data (but not necessarily to be built to accommodate very large amounts of data that are implausible).

Behavioural adequacy We want the model to be *incremental*, i.e. to improve its performance (however we choose to measure and evaluate performance) progressively with incoming data. While we want our model to rely on the surface form of the input, we do not acknowledge the necessity to limit the input analysis to be white-space based tokenisation: a more sophisticated model based on the identification of patterns or constructions in the input is as plausible as a naive one. We want our representation to allow for explicit inclusion of analysis results beyond the word by word sequences typically used as input to today's models.

Computational habitability We want the model to be evaluable and transparent, and manageable computationally in face of large and growing amounts of input data it is exposed to. We do not want it to make assumptions of a finite inventory of lexical items or expressions.

Explicit representation of features We want the model to allow exploration by the explicit inclusion of features of potential interest, without requiring expensive recalculation and reconfiguration of the model.

Context and Anchoring We want the model allow the inclusion of extra-linguistic data and annotations. Linguistic data is now available in new configurations,

collected from situations which allow the explicit capture of location, time, participants, and other sensory data such as biometric data, meteorological data, and social context of the author or speaker. These data are potentially of great interest e.g. to resolve ambiguities or to understand anaphor and deictic reference and should not be represented separately from the linguistic signal.

1.2 Theoretical basis

We base our work on linguistic data on a vector space model of distributional semantics, incorporating constructional linguistic items together with and similarly to the way we incorporate lexical elements.

Distributional semantics Distributional semantics is based on well-established philosophical and linguistic principles, most clearly formulated by Zellig Harris ([1968]). Distributional semantic models aggregate observations of items in linguistic data and infer semantic similarity between linguistic items based on the similarity of their observed distributions. The idea is that if linguistic items — such as e.g. the words *herring* and *cheese* — tend to occur in the same contexts — say, in the vicinity of the word *smörgåsbord* — then we can assume that they have related meanings. This is known as the *distributional hypothesis*. [13] Distributional methods have gained tremendous interest in the past decades, due to the proliferation of large text streams and new data-oriented learning computational paradigms which are able to process large amounts of data. So far distributional methods have mostly been used for lexical tasks, and include fairly little of more sophisticated processing as input. This is, to a great extent, a consequence of the simple and attractively transparent representations used. This paper proposes a model to accommodate both simple and more complex linguistic items with the same representation.

Vector space models for meaning Vector space models are frequently used in information access, both for research experiments and as a building block for systems in practical use at least since the early 1970's. [15] Vector space models have attractive qualities: processing vector spaces is a manageable implementational framework, they are mathematically well-defined and understood, and they are intuitively appealing, conforming to everyday metaphors such as "near in meaning." [17] The vector space model for meaning is the basis for most all information retrieval experimentation and implementation, most machine learning experiments, and is now the standard approach in most categorisation schemes, topic models, deep learning models, and other similar approaches, including this present model.

Construction Grammar The *Construction grammar* framework is characterised by the central claims that linguistic information is encoded similarly or even identically with *lexical items*—the words—and their *configurations*—the syntax, both being linguistic items with equal salience and presence in the linguistic signal. The parsimonious character of construction grammar in its most radical formulations [1, e.g] is attractive as a framework for integrating a dynamic and learning view of language use with formal expression of language structure: it allows the representation of words together with constructions in a common framework. For our purposes construction grammar gives a theoretical foundation to a consolidated representation of both individiual items in utterances and their configuration.

2 HYPERDIMENSIONAL COMPUTING

We present here the general framework for hyperdimensional computing, which has been used i.a. for modelling lexical meaning in language, and outline an extension of it to handle more constructional linguistic items.

The style of computing discussed here was first described by Plate [10, 11] and called Holographic Reduced Representation. The idea is to compute with high-dimensional vectors or *hypervectors* [5] using operations that do not modify vector dimensionality during the course of operation and use. We use 2,000-dimensional vectors in these demonstrations and experiments.

Information encoded into a hypervector is *distributed* over all vector elements, hence "holographic." Computing begins by assigning *random seed vectors* for basic objects. In working with text, for example, each word in the vocabulary can be represented by a seed vector, also called the word's *index vector* or random *label*. These seed vectors remain unchanged throughout computations. We may use two kinds of seed vectors consisting of 0s, 1s and -1s, *sparse* and *dense*. The elements of sparse vectors are mostly 0s, the dense vectors have no 0s. In both sparse and dense vectors, 1s and -1s are equally probable (our sparse seed vectors have 10 of each) and thus the vectors have mean = 0.

Representations of more complex objects are computed from the seed vectors with three operations. Two correspond to addition and multiplication of scalar numbers. *Addition* is ordinary vector addition, possibly weighted and normalized. *Multiplication* is performed elementwise, also known as the Hadamard product. The third basic operation is *permutation*, which reorders (scrambles) vector coordinates. The number of possible permutations is enormous.

One further operation on vectors measures their *similarity*. We use the cosine, with values between -1 and and 1. A vector is maximally similar to itself and yields a cosine = 1. Cosine = 0 means that the two vectors are orthogonal and appear to have no information in common. A system for computing with hypervectors also will need to include a memory for such vectors.

2.1 Computing with Hypervectors

The following is a somewhat more formal overview of properties of hypervectors used in this paper. They are readily seen in dense (seed) vectors $A, B, C, \ldots$ of equally probable 1s and -1s.

Distribution (dot product, cosine): Two vectors taken at random are *dissimilar*; they are approximately orthogonal—*quasiorthogonal*, cosine close to 0. The number of quasiorthogonal vectors grows exponentially with dimensionality.

Addition ($+$) of vectors produces a vector that is *similar* to the inputs, e.g.,

$$A + B + C \sim A$$

A sum of vectors is increasingly similar to the vectors it is a sum of if they are repeatedly added into it, and decreases with the number of dissimilar or unrelated vectors added into it. This provides a convenient way of collecting observational data for e.g. distributional semantics.

Multiplication ($*$) of vectors produces a vector that is *dissimilar* to the inputs, e.g.,

$$A * B \not\sim A$$

However, multiplication *preserves similarity*: the distance between $Q * A$ and $Q * B$ equals the distance between A and B.

A vector of ± 1s multiplied by itself produces a vector of 1s

$$A * A = \mathbf{1}$$

which means that the vector is its own *inverse*. That makes multiplication convenient for *variable binding*: variable x bound to value a—i.e., $\{x = a\}$—can be encoded by $X * A$, and the value can be recovered from the bound pair by multiplication:

$$X * (X * A) = (X * X) * A = \mathbf{1} * A = A$$

Multiplication *distributes over addition*, just as in ordinary arithmetic, because vectors are added and multiplied elementwise:

$$Q * (A + B + C + \ldots) = (Q * A) + (Q * B) + (Q * C) + \ldots$$

As a consequence, the value of x can be recovered approximately from the sum of several variable–value pairs, such as $\{x = a, y = b, z = c\}$:

$$\begin{aligned} & X * ((X * A) + (Y * B) + (Z * C)) \\ & = (X * (X * A)) + (X * (Y * B)) + (X * (Z * C)) \\ & = X + (X * Y * B) + (X * Z * C) \\ & = X + \text{noise} \\ & \sim X \end{aligned}$$

Elementwise multiplication is useful with dense vectors but not with sparse vectors because the product of sparse vectors is usually a vector of 0s, and because a sparse vector has no inverse.

Random *permutations* (Π_i) resemble multiplication: they produce a vector that is dissimilar to the input, they preserve similarity, are invertible, and distribute over addition; they also *distribute over multiplication*. Permutations provide a means to represent sequences and nested structure. For example, the sequence (a, b, c) can be encoded as a sum

$$\begin{aligned} S_3 &= \Pi(\Pi(A)) + \Pi(B) + C \\ &= \Pi^2(A) + \Pi(B) + C \end{aligned}$$

or as a product

$$P_3 = \Pi^2(A) * \Pi(B) * C$$

and extended to include d by $S_4 = \Pi(S_3) + D$ or by $P_4 = \Pi(P_3) * D$. The inverse permutation Π^{-1} can then be used to find out, for example, the second vector in S_3 or what comes after A and before C in P_3. If the pair (a, b) is encoded with two unrelated permutations Π_1 and Π_2 as $\Pi_1(A) + \Pi_2(B)$ then the nested structure $((a, b), (c, d))$ can be represented by

$$\begin{aligned} & \Pi_1(\Pi_1(A) + \Pi_2(B)) + \Pi_2(\Pi_1(C) + \Pi_2(D)) \\ & = \Pi_{11}(A) + \Pi_{12}(B) + \Pi_{21}(C) + \Pi_{22}(C) \end{aligned}$$

where Π_{ij} is the permutation $\Pi_i \Pi_j$.

The power of computing with numbers follows from the fact that addition and multiplication form an algebraic structure called a *field*. We can expect computing with (dense) hypervectors to be equally powerful because addition and multiplication approximate a field and are complemented by permutations that interact in useful ways with addition and multiplication.

3 HYPERDIMENSIONAL COMPUTING APPLIED TO LINGUISTIC DATA: RANDOM INDEXING

The operations presented above have been used in the *Random Indexing* approach to implement distributional semantics for language identification and to represent the meaning of words in a *word space model*.

3.1 Language Identification

In a recent experiment, high-dimensional vectors were used to represent properties of an entire text in a single *text vector* in order to identify the language it was written in. [4] The text vector of an unknown text sample was compared for similarity to precomputed *language vectors* assembled from processing known language samples. Character counts are known to be a fairly good indicator of language. [9] This model used frequencies of character sequences of length n—n-grams—observed in the text: as an example, the text "a book" gives rise to the trigrams "a b", " bo", "boo", and "ook". For arbitrary alphabets of L letters, there would be $(L + 1)^n$ n-grams to keep track of; in the case of English, which uses an alphabet of 26 letters (plus Space) this means keeping track of $273 = 19{,}683$ different trigram frequencies. These numbers grow quickly as the window size n increases.

In this experiment, each character was given a randomly generated index vector, and each window was represented by a (componentwise) product of its letter vectors permuted

according to their place in the window. For example, the trigram "boo" (from "book") was encoded into a window vector as $V_w = \Pi^2(B) * \Pi(O) * O$. The text vector is then formed by adding together the vectors for all the windows in the text:

$$\text{text\_vector} = \sum_{w \in \text{text}} V_w$$

The text vector for a text is then compared to previously similarly computed language vectors using cosine as a distance measure. The language whose language vector shows the highest cosine to the text vector is assumed to be the language the text sample is written in. In the experiment, tri- and tetragram yielded language identification accuracy (using the EUROPARL corpus) of more than 97%.

3.2 Lexical Semantic Space

To build a word space model to represent distributional data of words, a text sample is scanned one word at time. Each word in turn is the *focus word* of the scan with a *context window* of k preceding and succeeding words. When a word appears for the first time, it is assigned a randomly generated sparse *index vector* and an initially empty *context vector* of equal dimensionality. The index vectors of the words in the context window are *added* into the focus word's context vector. The resulting context vectors capture "bag-of-words" semantics of the focus word, reflecting their similarity of use. If the text sample is large enough, the similarity measure captures meaningful intuitions of term similarity.

Parameters for these models are most importantly word weighting, to assess how notable an observation of some word is or the size of the context, which ranges from entire "documents", such as in Latent Semantic Analysis [6, 7] to local contexts of the 2 to 3 closest words. A broader context captures topical or associative similarity of words where a more local context captures *paradigmatic* or replaceability relations such as synonymy or antonymy and *syntagmatic* combinability relation such as attributive or predicative relations. A window size of 2 or 3 words before and after a *focus word* has been found to yield good results as measured by synonym tests and other similar lexical semantic tasks [12] and various methods for weighting the observed occurrences have been tested to achieve on-line learning without becoming too sensitive to infrequent anomalous occurrences.

A step toward capturing structure has been taken by treating the words before the focus word differently from the words after. This is done by permuting their labels differently before adding them into the context vector. This use of permutation includes information about the preceding context and succeeding context separately, but allows for both to be used as an aggregate similarity measure for a word. [14] Several current neural network models use similar insights to represent sequential information.

3.3 Structure in linguistic data

The context vectors computed in random indexing reflect semantic similarity of words but do not easily allow for the inclusion of structural characteristics of an utterance, beyond what is observable from lexical statistics.

The operators described in Section 2 can be understood as a Vector Symbolic Architecture [2], and can be used to conveniently represent the many levels of abstraction in linguistic data. Suggestions to combine data in a tensor model [16, e.g.] are to some extent similar with respect to representational adequacy and power but are much more laborious computationally. A feature of interest can be included in the representation by overlaying it using addition and separated from others through permutation or multiplication, depending on how retrievable it is intended to be. A feature of interest can be included as a combination of other features, or independently of others.

Posit an utterance such as given in Example (1).

(1) Dogs chew bones.

This can be represented—as in typical search engine applications—as a combination of the vectors $\bar{v}$ representing the lexical items in play. Those representations can be random index vectors or context vectors or word embeddings obtained from previous analyses. The representation of the utterance can then most simply be achieved through simply adding the vectors for each participating lexical item together as in Equation 1.

$$\bar{v}_{dog} + \bar{v}_{chew} + \bar{v}_{bone} \tag{1}$$

Of course, in most practical applications, some informationally relevant scalar weighting of the items might be motivated.

$$w_{dog} \cdot \bar{v}_{dog} + w_{chew} \cdot \bar{v}_{chew} + w_{bone} \cdot \bar{v}_{bone} \tag{2}$$

This simple lexical representation can easily be extended to accommodate more elaborate information. Observing that the utterance in question is in present tense, we can simply add that information in by adding a vector which is randomly generated to represent that observable feature of the utterance. To collect tense information and keep it separate and separately retrievable from the lexical information, it can be permuted with a designated also randomly generated permutation for tense-related information. This information can be added in and later retrieved by similarity matching of vectors, using the $\prod tense$ permutation as a filter.

$$w_{dog} \cdot \bar{v}_{dog} + w_{chew} \cdot \bar{v}_{chew} + w_{bone} \cdot \bar{v}_{bone} + \Pi_{tense} \bar{v}_{present} \tag{3}$$

If it interests us to keep track of qualities of referents, we might want to add information about the morphological state of the word in question or the character of some of those referents. In Example 4 we could e.g. note that the referent to the agent of the clause ("dogs") is animate and that this referent is in plural number. This information can be added in and later retrieved by similarity matching of vectors, using a dedicated permutation such as Π_{agent} and vectors such as $\bar{v}_{animate}$ and $\bar{v}_{plural}$ together with $\bar{v}_{dog}$, the representation of dog.

$$w_{dog} \cdot \bar{v}_{dog} + w_{chew} \cdot \bar{v}_{chew} + w_{bone} \cdot \bar{v}_{bone} + \Pi_{tense}\bar{v}_{present} + \Pi_{agent}(\bar{v}_{animate} + \bar{v}_{plural} + \bar{v}_{dog}) \quad (4)$$
$$\Pi_{location}(41°24'5''N2°9'23''E)$$

This principle enables the representation to harbour many types of information regardless of how it has been generated or extracted from an utterance, but all aggregated in the same hyperdimensional space to be used for downstream processing in ways that the representation does not need to take into account at perception time.

4 QUANTITATIVE CHARACTERISTICS

The general task for a knowledge representation is to provide *features* which with to represent observed characteristics of interest of some situation in the world, to aggregate such observable features of some situation of interest into a represented *state*, to allow further processes to *verify* if an observation has been recorded in that state, and to *decompose* a state representation into the separate features which have composed it.

The advantage of using a holographic rather than a localist model is that for a d-dimensional representation, where a one-hot model allows for d features with lossless aggregation and retrieval from a state, the variation space of the hyperdimensional approach affords by virtue of the random patterns, permutations, and multiplications a vastly larger feature palette. This makes possible, as shown above in Section 3, the representation of an entire vocabulary and their cooccurrence statistics to be handily accommodated in a 2,000-dimensional space.

The aggregation of features into a state is done simply by vector addition, occasionally using a permutation to separate aspects of a feature. Verifying if a feature is present in a state vector is done using most simply by a dot product or a cosine similarity measure. The choice of d, the dimensionality of the representation, determines the capacity of the space. As can be expected, a larger dimensionality allows greater capacity: a 100-dimensional space can store less information, i.e. fewer distinct features, for each state than a 2,000-dimensional does. If we wish to aggregate N (near)-orthogonal features by addition into a state vector, their relative cosine distance to that resulting state vector will be $\sqrt{(}1/N)$. The expected size of N determines how large d must be chosen to be to ensure that the cosine is at a safe margin from the noise threshold occasioned by the randomisation procedure. If a state vector is expected to hold on the order of 100 unweighted feature vectors, the a resulting relative cosine between each feature vector and the state vector will be 0.1 on average. In a 1,000-dimensional space, this is about three to four times the noise threshold; in a 2,000-dimensional space about five to six times from the noise threshold. The graphs in Figure 1 show how the noise threshold compares across some typical dimensionality settings.

The size of the representation does not grow with the number of feature vectors aggregated into the state vector, except by density of the state vector. Neither does the number of potential features — the size of the lexicon and the combined size of all potentially interesting constructions — occasion more than linear growth for the system in its entirety.

5 EMPIRICAL VALIDATION

We intend to validate our approaches by applying selected technologies to *tasks* which require understanding of linguistic content. Typical tasks we exepect to address are more advanced language understanding tasks such as *authorship and genre identification*, *author profiling*, *attitude and sentiment analysis*, *viewpoint analysis*, and *topic detection and tracking* as well as some more theory-internal tasks such as *semantic role labeling*. The tasks we are interested and where holographic representations are most obviously useful are such where a broad range of features is necessary to perform well in them, and where performance is constrained by the challenge of incorporating information on several levels of abstraction simultaneously in an integrated processing model.

Current experimentation with this model at present is focussed on authorship profiling which relies on features beyond the lexical and on tracking social media posts on meteorological events which is highly temporally determined and locational in nature.

A simple sentence level task demonstrated here is that of question classification: assigning a category to a question which will impose a semantic constraint on the answer. Example categories are "Human, individual", "Number, date", "Location, city" [8]. This task forms a basis for many question answering systems, and a fair amount of effort has been put into optimising performance in it. The most important features for this task, as for many language tasks, are lexical: the words "How long" in the question "How long is the Coney Island boardwalk?" indicate that the expected answer is a number, such as a distance or a time period. Even fairly naive word frequency methods achieve around 50% accuracy and with some multi-word terms and dependencies added, optimised systems yield at least 80% accuracy.

We do not attempt here to push the envelope for this data set — this task is today considered to be solved and superseded by more elaborate tasks in question answering. We use this to demonstrate how our representation allows the addition of more information in a fixed dimensionality without perturbing simpler models in the same representation.

We used a set of 5,500 previously labelled questions to train a model which incorporates all words in the model together with some semantic roles those words enter into. We selected a small set of roles which we expect to be of interest for the purpose of inferring the target entity of the question: the question word itself (Who, How, What, etc), the tense of the question main verb, the subject of the main verb, any other clause adverbial, and the binary feature of negation or not. The labels were given on two levels of granularity, coarse-grained ("Human", "Location", "Number", "Description",

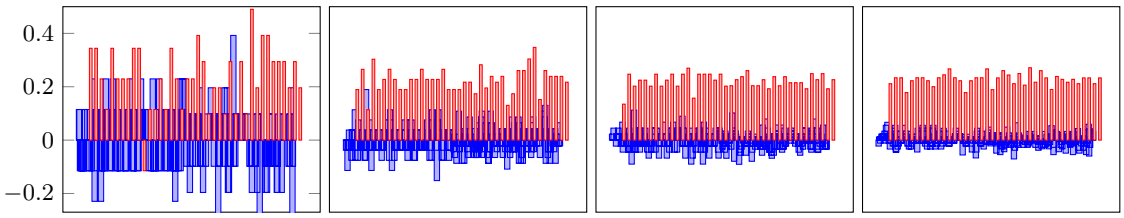

Figure 1: Cosine of feature vectors to state vector compared to random unrelated vectors in 100, 500, 1,000, and 2,000 dimensions

"Entity", "Abbreviation") and fine-grained (47 subcategories of the coarse-grained categories). In one condition, each label was given a state vector with each of the words from every question it has been applied to; in another, semantic roles were added together with the words that participated in them, in each case permuted for the role in question; in a third condition both were added together. A test set of 500 questions were then used to evaluate the models, by building similar vectors for each question and selecting the label with the greatest cosine similarity to it as a candidate label. This was done using

The state vector of a question is then composed additively of the bag of words of surface tokens, and for each role of interest, the lemma form of the word permuted by a role-specific permutation. Each category is represented by a sum of all state vectors for questions in the training set labeled with that category. Each test item is then used in three conditions: a state vector formed from only bags of words, one with only roles, and one with both. These then matched to the category vectors, and the category whose vector shows the closest cosine distance is used for evaluation. This was done for the three conditions: lexical items only, semantic roles only, and the combination of the two. The wlexical items-only condition was tested both on a semantic space built using lexical items only and one using both lexical items and semantic roles. Table 5 shows the results, most notably that the disturbance from the more elaborate model does not appreciably change the results from the simpler model: in this example only a slight reranking of three questions in the output for the lexical model caused it to lose 0.6 percentage points of precision while allowing the more sophisticated representations holographically concurrently with it.

	coarse-grained (6 categories)		fine-grained (47 categories)	
	av rank correct label	accuracy by %	av rank correct label	accuracy by %
	Words-only space			
words only	1.91	60.2	4.84	57.4
	Combined semantic space			
words only	1.92	60.2	5.88	56.8
semantic roles	1.91	58.0	6.78	60.2
roles + words	1.85	66.2	4.64	62.4

Table 1: Question classification results, average rank of correct answer and percentage of items with correct answer at rank 1

6 CONCLUSIONS AND RELATION TO OTHER APPROACHES

Human language has a large and varying number of features, both lexical items and constructions, which interact to represent referential and relational content, communicative intentions of the speaker, situational references, discourse structure, and many others. A hyperdimensional representation is eminently suitable for representing language, to aggregate and handle the wealth of linguistic data and its range of each in themselves weak features. It also seamlessly accommodates information from outside the linguistic signal in the same representation.

A hyperdimensional representation can work in conjunction with other representations: it can act as a bridge between symbolic and continuous models, by accepting symbolic data and thus be to serve as an encoder for e.g. neural models, embeddings-based models, or other approximative classification schemes. It allows the seamless and explicit addition to and recovery from a representation of arbitarily complex and abstract features. A model with an accessible symbolic representation together with continous data allows for choice and preference to be represented simultaneously, and allows for objective functions for learning to be represented on levels of abstraction salient and relevant to task at hand.

In addition, the memory footprint of a hyperdimensional model based on random indexing is habitable: an overly large amount of data leads to saturation of the model and a graceful degradation of performance, not to memory overflow. We expect to see how models based on this approach can be used not only for experimentation but, since their designed is principled and based on generality and accessibility, for application purposes.

REFERENCES

[1] William Croft. 2005. Radical and typological arguments for radical construction grammar. In *Construction Grammars: Cognitive grounding and theoretical extensions*, Jan-Ola Östman and Mirjam Fried (Eds.). John Benjamins, Amsterdam.

[2] Ross W Gayler. 2004. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059* (2004).

[3] Zellig Harris. 1968. *Mathematical structures of language*. Interscience Publishers.
[4] Aditya Joshi, Johan T Halseth, and Pentti Kanerva. 2016. Language geometry using random indexing. In *International Symposium on Quantum Interaction*. Springer, 265–274.
[5] Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation* 1, 2 (2009), 139–159.
[6] Pentii Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Cognitive Science Society*, Vol. 1.
[7] Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review* 104, 2 (1997).
[8] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics (COLING)*. International Committee for Computational Linguistics.
[9] Seppo Mustonen. 1965. Multiple discriminant analysis in linguistic problems. *Statistical Methods in Linguistics* 4 (1965), 37–44.
[10] Tony A Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural networks* 6, 3 (1995), 623–641.
[11] Tony A Plate. 2003. *Holographic Reduced Representation: Distributed representation for cognitive structures*. Number 150 in CSLI Lecture notes. CSLI Publications.
[12] Magnus Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD Dissertation. Department of Linguistics, Stockholm University.
[13] Magnus Sahlgren. 2008. The distributional hypothesis. *Rivista di Linguistica (Italian Journal of Linguistics)* 20 (2008), 33–53. Issue 1.
[14] Magnus Sahlgren, Anders Holst, and Pentti Kanerva. 2008. Permutations as a means to encode order in word space. In *The 30th Annual Meeting of the Cognitive Science Society (CogSci'08)*.
[15] Gerard Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620. DOI:http://dx.doi.org/10.1145/361219.361220
[16] Fredrik Sandin, Blerim Emruli, and Magnus Sahlgren. 2017. Random indexing of multidimensional data. *Knowledge and Information Systems* 52, 1 (2017), 267–290.
[17] Hinrich Schütze. 1993. Word Space. In *Proceedings of the 1993 Conference on Advances in Neural Information Processing Systems, NIPS'93*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 895–902.