High-dimensional vector spaces can accommodate constructional features quite conveniently

Jussi Karlgren
Numolo
Stockholm, Sweden
jussi@lingvi.st

Abstract

Current language processing tools presuppose input in the form of a sequence of high-dimensional vectors with continuous values. Lexical items can be converted to such vectors with standard methodology and subsequent processing is assumed to handle structural features of the string. Constructional features do typically not fit in that processing pipeline: they are not as clearly sequential, they overlap with other items, and the fact that they are combinations of lexical items obscures their ontological status as observable linguistic items in their own right. Constructional grammar frameworks allow for a more general view on how to understand lexical items and their configurations in a common framework. This paper introduces an approach to accommodate that understanding in a vector symbolic architecture, a processing framework which allows for combinations of continuous vectors and discrete items, convenient for various downstream processing using e.g. neural processing or other tools which expect input in vector form.

1 Continuous and discrete models

Processing models and memory models for knowledge-intensive tasks of many kinds are currently implemented as vector models of various kinds. The latest few generations of implemented natural language processing tools follow this trend, and benefit from the convenient and well-understood processing framework geometric models offer, the seamless incorporation of learning into a continuous model, and the attendant possibility to generalise from a large body of background knowledge to work with a specific task. Results on benchmark tasks has been impressive, and this is gratifying for those of us who have advocated for unsupervised learning, for statistical and probabilistic approaches, and for somewhat neurophysicologically inspired processing architectures.

The most obvious flip side of the impressive results comes with the cost of running such models. The amount of training data required is enormous compared to previous generations of models, the number of parameters to set during training is orders of magnitudes of orders of magnitudes larger, and the expense for appropriate computing infrastructure is prohibitive.

Much of this computing effort appears to be wasteful for those who have an understanding of the linguistic signal. The processing model starts from no understanding of what it is expected to model, is fed chunks of linguistic data with the instruction to pay attention to character sequences (mostly but not always with special attention paid to white space and sentence separators), and eventually will be able to relate the strings it has been fed with to each other in interesting and behaviourally adequate ways.

Previous generations of statistical models have at times experimented with including lexical categories or structural features to enrich the string input, but results have been equivocal and the current generation of natural language processing tools has dispensed with dependency graphs and lexical classes, preferring to infer the operatively effective relations between string elements directly from the data.

2 One can have both

For a linguist, the entire approach causes some frustration. One view of a linguist's job description is to provide the appropriate features for a learning system to pay attention to, and to strive to optimise the convergence of features, categories, and dimensions of variation for a natural language processing system to deliver the best results, the steepest learning curves, and the smallest set of parameters for some set of tasks. The current generation of natural language processing tools do not invite such intervention or such hypothesis testing: indeed,

Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023), pages 31 - 35

the practice of feature engineering is somewhat frowned upon.

In more general terms, a continuous geometric model has intuitively appealing qualities (even to the extent that the metaphors such a model invites coupled with our understanding about geometry in a physical world can lead our intuitions astray (Karlgren and Kanerva, 2021)): where a symbolic model allows for greater transparency, greater explanatory power, convenient inspectability and editability, and a more direct path to hypothesis testing, a continuous geometric model offers robustness, coverage, and generalisability. And today, by representing linguistic observations as vectors we will have a convenient interface to downstream computation using various past, current, and most likely many future flavours of machine learning.

There is no inherent contradiction between continuous models and discrete elements of study. Words are discrete observations and are routinely represented as vectors in language processing tasks, typically encoded by a shallow neural net which takes local context into account in narrow windows to model syntactic dependencies or slightly larger windows to model topical association. Configurational features such as elements from construction grammars can be represented similarly.

3 The general idea of high-dimensional computation

High-dimensional computation allows for the incorporation of linguistic items, single lexical items, configurational elements, or constituent structures jointly, using simple operations. This framework was first introduced by Plate under the name Holographic Reduced Representation (HRR; Plate, 1991, 2003) and further developed as Multiply–Add–Permute (MAP: Gayler, 1998), Vector Symbolic Architecture (VSA: Gayler, 2004), and Hyperdimensional Computing (Kanerva, 2009). The idea is to encode information in a vector with three simple linear algeabric operations that keep vector dimensionality constant: *vector addition*, *vector multiplication*, and *permutation*. *Addition* of two vectors yields a new vector *similar* to its operand vectors; addition can be used to represent a set. *Multiplication*, coordinate by coordinate, yields a product vector which is *dissimilar* to its operand vectors. *Permutation* takes a single vector, rearranges its coordinates, and produces a vector that is *dissimilar* to the operand. The operations are *invertible*: the operations can be undone and the component vectors retrieved from the result. These operations are based on a well-understood computational algebra, similarly to how most vector models rely on geometry. A vector space together with linear algebraic manipulation operations and geometric access and analysis operations can be used to combine observations into a common vector representation systematically, transparently, and explicitly, and for our purposes allows us a convenient way to evaluate the information value of features we expect to be important to understand the linguistic signal.

Random indexing is a high-dimensional computation framework, which assigns randomly generated fixed-dimensional index vectors to observations of interest, to be combined using the above operations. Random indexing traces its roots to Kanerva's Sparse Distributed Memory framework (Kanerva, 1988). A randomly generated index vector is defined to be sparse, i.e. to mostly contain 0s with a small number of 1s and -1s—say 10 non-zero elements in a 1000-dimensional vector—and the characteristics of high-dimensional spaces are such that two such randomly generated vectors will be very close to orthogonal. We assign random index vectors to each linguistic item of interest: single words and constructions alike. If a new previously unencountered item shows up during processing, it can be assigned a new unique index vector without retooling the previously known space of items. In random indexing of linguistic material, addition is used to combine observations that are collocated into a joint vector: an utterance can be represented as the sum of index vectors for every word in it. Permutation can be used to distinguish item occurrences in different roles or different surface forms, to distinguish cases, semantic roles, or head-attribute relations, e.g. This framework will allow us to represent sequences and configurations together with their constituent elements conveniently.

4 Some examples

This section is based on some previously published example implementations and experiments (Karlgren and Kanerva, 2019; Karlgren et al., 2018). Previous work using this approach was used to build general associative lexical resources (Sahlgren et al., 2016) using permutations to differentiate between left hand and right hand context (Sahlgren

et al., 2008). This model is a parsimonious method to aggregate cooccurrence statistics and has proven quite useful in practical application, but has today been superseded by large language models that are able to capture longer range dependencies.

This purely lexical model can be enhanced by adding constructional elements, inflectional information, semantic roles, and even contextual extralinguistic information. Sentences such as the ones given in Example (1) both involve fish. This is of course for some purposes a notable observation, but we can add the observation that the fish in question participate in different roles in the sentences. This can be encoded by adding a semantic role label to the *fish* vector, adding a tense and aspect annotation to the main verb or even to all referential expressions to indicate that fish in Sentence (1-b) are agents doing their thing in present progressive in contrast with the fish in Sentence (1-a). What features to represent are up to the hypotheses being considered, and adding spurious features will not confuse the system except adding a slight noise to the resulting vector. The features of interest can be retrieved separately, since the operations used are invertible. The representation of $fish_{agent}$ can be derived from the representation of $fish$ by invoking a permutation specific for agent roles.

(1) a. The fishermen have cured the [fish]$_{patient}$ by smoking or by salting them in brine.
b. The [fish]$_{agent}$ are jumping up like birds now.

(2) a. $fishermen + fishermen_{agent} + cure + fish + fish_{patient} + present - perfect + smoke + salt + brine + ...$
b. $fish + fish_{agent} + jump + present - progressive + birds + now + ...$

In a series of experiments on a commercial data set of customer reviews we tested the effect of adding amplifiers, negations, and constructional markers for attitudinal expression in addition to lexical features. We represented each sentence in the collection as a vector into which we added the index vector for each term present in it, weighted by inverse frequency. We also added a separate $amplifier$ vector if an amplifier was present; a $negation$ vector if the main verb of the sentence was negated; a number of verbal class vectors $cogitation$, $expression$, $privatesensation$, and some others; separate vectors for each observed tense form in the sentence; vectors for presence of a personal pronoun; vectors for a number of attitudinal classes; and a series of constructional vectors for presence of subclause, presence of auxiliaries, and presence of adverbial constructions. These vectors form a lexical-semantic-constructional space with all features represented jointly. To demonstrate how this space can be queried for features, we represented the Sentence (3-a) separately with only lexical features and with only semantic and constructional features. We then retrieved the most similar sentence from the joint vector space: the lexical vector retrieves Sentence (3-b); the semantic and constructional vector retrieves the more personally expressive utterance with negative sentiment in Sentence (3-c). The original objective of this experimentation was to find attitudinal expressions for certain types of product: here it is useful to show how a large number of features can be used to build a space and then that space can be queried with attention paid to subsets of features.

(3) a. I really did not like the clarinet, I am afraid: it sounded weak!
b. My sister plays the clarinet.
c. I'm surrounded by really soft decadent pillows which do not work for me at all.

In a continued set of experiments on attitude analysis we experimented with constructional features and their distribution in a dataset of some one million microblog posts that mention among other things corporate entities. We represented each microblog post as a sum vector of index vectors for individual lexical items; for unique triples of part-of-speech tags—each triple such as $DT - JJ - NN$ or $VBD - RB - RB$ having been assigned its own index vector; for observed tense and aspect for the main verb; for observations of the presence of modal auxiliaries and various adverbs; for semantic role labels for the agent of each clause; for the presence of several categories of amplifiers; and for some extracted configurations for verbs of utterance and cogitation, e.g. $utteranceverb - that$ and other frequent constructions. These features were extracted using the NLTK toolkit (Bird, 2006) and on lexical resources coded using a comprehensive lexically oriented grammatical description of English (Quirk et al., 1985).

Similarly to the Example (3) above, we found, as shown in Example (4) that the resulting representa-

tion if only tested for lexical content indicated that the most similar utterance to Sentence (4-a) would be Sentence (4-b) while incorporating the various constructional features Sentence (4-c) was found to be the best match. This we found to be useful for a project on tracking corporate sentiment online (Karlgren et al., 2012). In this case, the agency and animateness of the corporation in question is the significant feature linking the first and the third utterance.

(4) a. <CORPORATION X> announced their intention to move their corporate headquarters to Houston.
b. Houston is rocking, wow <CORPORATION X>!
c. <CORPORATION X> plans to comply with the court order and will provide information to A!

Our examples taken from previous implementations presented here are intended to demonstrate that with very simple additional processing, constructional features can be added to a processing model, allowing the concurrent inclusion of many information sources into one joint representation in a computationally and conceptually habitable way, well supported by established computational practice. These operations are low effort, and can be done selectively to provide a test bench for experimentation to find what the relative effect of constructional features are, given e.g. a classification task or a ranking problem. This information can be selectively retrieved from the vectors as shown in the above examples, but more importantly, they can be used to enrich some given lexical model with constructional features of choice by the operations outlined above. They are not intended to convince the reader of the utility of any specific set of features or of their suitability to some established task—it is the processing model and computational approach that is novel and useful, not our hypotheses about language and its functions.

Such resulting vectors are similar to pre-trained off-the-shelf word embeddings such as are routinely used as input to downstream machine learning models and indeed constructional features can be combined with such word embeddings through the application of addition operations. In general, the current generation of natural language processing tools are agnostic to the content of their input and are able to accommodate even weak signals found in the input. This suggests that a useful validation path to test hypotheses of the effectiveness and usefulness of constructional features is not only to study the end results on benchmark tasks, but the path to get to those results, and to investigate how the model takes the potentially enriched information into account in its training.

5 Take home

This short paper attempts to convince its readers that it is possible to combine lexical and configurational features in a joint vector space model; that combining configurational or constructional features and lexical features in a joint vector space model is a useful and desirable path to investigate the validity of constructional hypotheses; that radical construction grammars provide a theoretical back end to such representations; and that hyperdimensional computing or vector symbolic architectures provide a well-established computational framework for processing such discrete information into a form which can be ingested by today's most popular processing tools. The details of our implementation are not important for this argument but random indexing is a convenient and lightweight approach to combining heterogenous information into a geometric representation.

References

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*.

Ross W Gayler. 1998. Multiplicative binding, representation operators & analogy. In D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*. New Bulgarian University.

Ross W Gayler. 2004. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *arXiv:cs/0412059*.

Pentti Kanerva. 1988. *Sparse distributed memory*. MIT press.

Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1(2).

Jussi Karlgren, Lewis Esposito, Chantal Gratton, and Pentti Kanerva. 2018. Authorship profiling without using topical information: Notebook for PAN at CLEF. In *19th Working Notes of CLEF Conference and Labs of the Evaluation Forum, CLEF 2018, Avignon, France*, volume 2125. CEUR-WS.

Jussi Karlgren and Pentti Kanerva. 2019. High-dimensional distributed semantic spaces for utterances. *Natural Language Engineering*, 25(4).

Jussi Karlgren and Pentti Kanerva. 2021. Semantics in High-Dimensional Space. *Frontiers in Artificial Intelligence*, 4.

Jussi Karlgren, Magnus Sahlgren, Fredrik Olsson, Fredrik Espinoza, and Ola Hamfors. 2012. Usefulness of sentiment analysis. In *Proceedings of the European Conference on Information Retrieval (ECIR)*.

Tony Plate. 1991. Holographic Reduced Representations: Convolution Algebra for Compositional Distributed Representations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Tony A Plate. 2003. *Holographic Reduced Representation: Distributed representation for cognitive structures*. Number 150 in CSLI Lecture notes. CSLI Publications.

Randolph Quirk, Sidney Greenbaum, Geoffrey Neil Leech, and Jan Svartvik. 1985. A Comprehensive Grammar of the English Language.

Magnus Sahlgren, Amaru Cuba Gyllensten, Fredrik Espinoza, Ola Hamfors, Jussi Karlgren, Fredrik Olsson, Per Persson, Akshay Viswanathan, and Anders Holst. 2016. The Gavagai living lexicon. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*.

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)*.