Hierarchical Semantic Composition Using High Dimensional Vectors and Random Indexing

Catherine Zeng [1] Bob Berwick [1]

[1]Massachusetts Institute of Technology

Background

Modern developments in natural language processing are still bad at doing anything that involves the hierarchical structures that people use, with most neural models such as recurrent neural networks (RNNs) being linear from end-to-end. A good language representation model should be capable of encoding hierarchical linguistic structure. However, bag-of-words semantic vector representations like LSA and Word2Vec fail to encode any structure outside of word order. Further, the Word2Vecs views finding appropriate vector representations as an optimization problem, is exceptionally computationally expensive, and introduces a batch process into the data flow.

Using random indexing and high-dimensional, sparse distributed vectors [2] enables many features that align with linguistic intuition: concepts can be connected by short links in semantic space because all vectors are quasiorthogonal, there is a high degree of tolerance for noise, and analogical reasoning is possible through using Holographic Reduced Representations.

Further, there has been increasing evidence for the existence of high-dimensional, sparse binary vectors in biological organisms. For example, in fruit fly olfactory circuits, 50-dimensional odorant receptor neurons (ORNs) in the fly's nose are connected to neurons that project to 2,000 Kenyon cells by sparse, binary random connection matrices [1]. It has also been shown that the cerebellum uses sparse, dimensionality expanding encoding, with large numbers of granule cells receiving few inputs from "mossy fibers" and Purkinje cells receiving tens of thousands of inputs from "parallel fibers" and a "climbing fiber".

Method

In order to associate compositional structures in distributed representations, convolution algebra is used to form Holographic Reduced Representations [5] of word embeddings.

Symmetry preservation in convolutions allows the degree of similarity between two vectors to be conserved when both are bound with the same binding. For example, if vectors for "red" and "green" are similar, then the bindings $red \oplus apple$ and $green \oplus apple$ would also be similar by approximately the same degree. This symmetry preservation property is important because it allows us to draw analogies between two word vectors based on similarity of fields.

Superimposing large numbers of bindings can cause inference that increases the amount of noise resulting vectors have. Interference can be mitigated by increasing the number of dimensions used for the binary vector.

Encoding & Decoding

- **Encoding** The encoding process uses superposition and binding to bind vectors into a "memory trace", using addition and xor respectively. For example, the pattern $(yellow \oplus banana) + (red \oplus apple)$ can be used to represent the composition of two pairs "yellow-banana and red-apple".
- **Decoding** The decoding process applies inverse convolution operations on a memory trace and a single field to return a noisy version of what was associated with the field. For example, $yellow^{-1} \oplus yellow \oplus banana \approx banana$. In our case, the xor operator is invertible (xor is its own inverse). Because the result is noisy, the dot-product is then used to find its the closest association in memory space.

Generating Word Embeddings

Newly encountered words are generated a d dimensional random, binary vector containing $-1s$ and $1s$ (using -1 and 1 lets us use the multiplication operator for xor) that is stored as an environment vector e_i. Each newly generated word additionally is given a memory vector m_i that is the linear combination of the environment vectors of surrounding words multiplied by their part of speech s_i and their structural relationship r_i (each of which is also symbolically represented by a d dimensional random, binary vector of $-1s$ and $1s$).

The structural relationship r_i is encoded as the movements needed to move from one word to another on the parse tree hierarchically. For example, the movement required to move from one leaf node to an adjacent leaf node when they have the same parent would be "up, down", which would be encoded as "10". Similarly, if moving from one word to another requires moving up the tree twice and then moving down three times, then the movement would be encoded as "11000". These movements are symbolically represented by d dimensional random, binary vectors consisting of $-1s$ and $1s$.

The word embedding rules are summarized as follows:

$$c_j = \sum_{i=1}^{n} e_i \oplus s_i \oplus r_i \tag{1}$$

$$m_i = m_i + c_j \tag{2}$$

Analogous Structures

The symmetry preservation property in convolutions allows us to apply analogical reasoning using the word embedding structure. For example, if we want to find "m_1 is to s_2 as m_2 is to what?" given the following word embeddings:

$$m_1 : s_2 :: m_2 :?$$

$$m_1 = (e_1 \oplus s_1 \oplus r_1) + (e_2 \oplus s_2 \oplus r_2)$$

$$m_2 = (e_3 \oplus s_1 \oplus r_1) + (e_4 \oplus s_2 \oplus r_2)$$

If we multiply the two memory vectors together, we get:

$$F_{m_1,m_2} = (e_1 \oplus e_3) + (e_2 \oplus e_4) + noise$$

Taking this compositional vector, if we further multiply F_{m_1,m_2} by s_2 to complete our analogy, we get:

$$F_{m_1,m_2} * s_2 = noise + e_4 + noise$$

Finally, we use the dot product to find the closet environment vector e_4 to complete the analogy that $m_1 : s_2 :: m_2 : e_4$.

Experiment

We used random indexing and $100,000$ dimensional vectors to generate word embeddings from the Penn Treebank. Once the vector representations have been generated, we probe representations for analogies in the form "a is to b as c is to what?" ($a : b :: c :?$) by multiplying semantic vectors together as described in the method section: $a * b * c =?$. Retrieving each vector used in the multiplication from memory space takes $O(1)$ time. Once the noisy resultant vector is found, using the dot product to compare it against stored word embeddings runs in $O(n)$ time with n being the number of words stored. All this we were able to do through direct manipulation of the word embedding representative structure, entirely online and without any training process.

For the purpose of reproducibility, we've release our implementation at github.com/yczeng/semantic-composition.

Results and Discussion

Gold Standard Parse Trees for 1,000 sentences from the British National Corpus were used to generate word embeddings. Some results are shown in the table below:

m_1	s_2	m_2	?
shoes	socks	mosses	ferns
canoes	fleet	elephants	herd
resonant	frequency	magnetic	field
basked	glow	luxuriating	feel

Table 1: Analogy where m_1 is to s_2 as m_2 is to what?

Unlike existing methods for generating word embeddings such as Word2Vec that uses a batch training process, using high dimensional vectors and random indexing is computationally inexpensive, online, and robust against noise. However, these word embeddings remain a static representation, so no learning is going. It would be interesting to see how these word embeddings could interact in a dynamic manner to self-organize and update.

References

[1] Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha.
A neural algorithm for a fundamental computing problem.
Science, 358(6364):793--796, 2017.

[2] Pentti Kanerva.
Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors.
Cognitive Computation, 1(2):139--159, 2009.

[3] Thomas K. Landauer and Susan T. Dumais.
A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.
Psychological Review, 104(2):211--240, 1997.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean.
Distributed representations of words and phrases and their compositionality.
In Advances in Neural Information Processing Systems, pages 3111--3119, 2013.

[5] T. A. Plate.
Holographic reduced representations.
IEEE Transactions on Neural Networks, 6:623--641, 1995.