Word Embeddings with HD Computing/VSA

Ryan Moughan

About Me

- Berkeley ‘20 BA in Computer Science and Statistics
- Berkeley ‘21 MS in EECS
- Did research at the Redwood Center from 2019 - 2021
- Currently a machine learning researcher at LLNL

This Week

- Motivation
- Background
- Hyperdimensional Word Embeddings
- Applications
- Questions

Motivation

Language

- What problem do we want to solve?
  - “I want a human robot!”
  - “I want a computer that understands language!”
  - I want to be able to increase ad revenue by 6% using natural language processing
  - I want to be able to replace 90% of the work of lawyers by being able to automate the parsing and creation of contracts
- “Language” is a rather big idea to try to handle
- Many people, including myself, try to jump in without thinking hard about this

“It seems that we need to know what it is we want to represent before we can start thinking about how to represent it in a computer system. Succinctly, it seems as if the recourse to semantics demands an explanation of the meaning of ‘meaning’.”

-Karlgren and Sahlgren

Karlgren, J., & Sahlgren, M. (2001). From words to understanding.

How to approach the problem

- If we want a computer to understand language, there are a couple of possibilities
  - Use vocalizations as input
  - Use images of books as input
  - Could even use Braille
- These all are more similar to how humans get our input, but they're not natural for computers (many are ML fields of their own)
- Let's try to simplify the problem
  - Skip the visual part of reading and just directly input the characters
  - Important question: are we losing anything with this approach?

Background

Understanding the Problem

- Say we have all of the text to Frankenstein and want to “learn” English
- What do we want to do with this?
- What can we do with this that is computationally feasible?

Distributional Semantics

- Distributional Semantics hypothesis: Words that occur in similar contexts have similar meanings
- But what exactly is a context?
  - Document
  - Paragraph
  - Sentence
  - Fixed window size of words

"These different contextual representations capture different types of semantic similarity; the document-based models capture semantic relatedness (e.g. "boat" – "water") while the word-based models capture semantic similarity (e.g. "boat" – "ship"). This very basic difference is too often misunderstood."

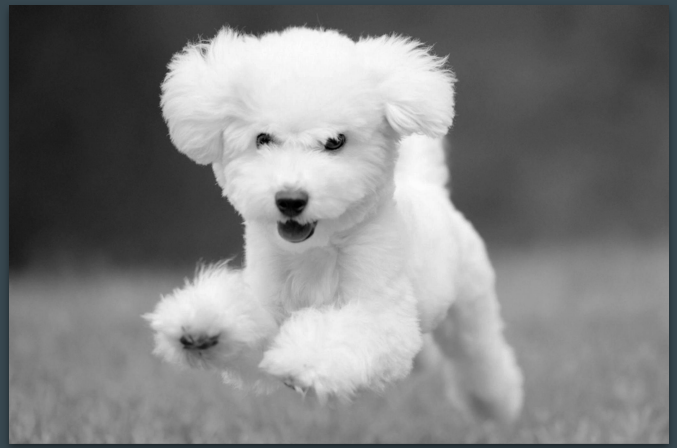
https://www.gavagai.io/text-analytics/a-brief-history-of-word-embeddings/

Distributional Semantics

- What does this look like?

"A white dog went for a walk."

- "dog" can be understood as a function of "A", "white", "went", and "for"
- This kind of seems useful?



Distributional Semantics (on steroids)

This dog helped him watch the sheep.

Then let's have a seat and watch the puppy dog parade.

Betsy held on like a dog on a bone.

Whenever it is possible, my dog accompanies me on a walk or ride or sail.

The shepherd and his dog could not keep them together.

The Great Pyrenees dog had adopted the buffalo calves shortly after they were born.

Distributional Semantics

- Over thousands of uses, maybe we can get something out of this?
- What do people think of this?
- Pros:
  - Simple approach that computers can understand
  - Good for basic semantic similarity
- Cons:
  - Limited scope of the “meaning of meaning”
  - One-dimensional view of language
  - Fixed window size can miss long-term dependencies

Hyperdimensional Word Embeddings

From Distributional Semantics to HD Word Embeddings

- With the distributional semantics hypothesis, we now have a straightforward way to define the meaning of a word
- How do actually go about computing this quantity though?
- Let's do a quick review of the HD operators

Brief Review of HD Operators

- Binding: $\mathbf{x} \oplus \mathbf{y}$
  - Creates a hypervector that is dissimilar to both inputs
  - Used to bind hypervectors together
  - Invertible and distributive
- Bundling/Superposition: $\mathbf{x} + \mathbf{y}$
  - Creates a hypervector that is similar to both inputs
  - Used to learn associations or form sets
- Permute: $\rho(\mathbf{x})$
  - Creates a hypervector that is dissimilar to the input
  - Used to encode structure
  - Invertible and distributive

From Distributional Semantics to HD Word Embeddings

- How can we use distributional semantics and HD operators to compute an embedding?
- Consider the following sentence:

"A white dog went for a walk."

- Say we want to find a word embedding for "dog"
- Using the distributional semantics hypothesis, we can just consider "dog" in terms of "A" + "white" + "went" + "for"

Context Embeddings

- These are called the *context embeddings*, denoted as [word]
- So for the previous sentence with a window size of 2:

"A white dog went for a walk."

- [dog] = A + white + went + for
- But are we losing information here? What about the relative order of words?

Order Embeddings

- That's why we have *order embeddings*! We denote these as <word>
- Here the permutation operator is able to save the day
- Using the same setup as before:

"A white dog went for a walk."

- <dog> = $\rho^{-2}(A) + \rho^{-1}(white) + \rho^{1}(went) + \rho^{2}(for)$
- Now we have a way to differentiate where a word occurred!

Order Embeddings

- Why are these called order embeddings?
- Say we wanted to recover the word that is most associated with being right before dog
- What we can do is:

$$\rho^1(< dog >) = \rho\left[\rho^{-2}(A) + \rho^{-1}(white) + \rho(went) + \rho^2(for)\right]$$

$$= \rho^{-1}(A) + white + \rho^2(went) + \rho^3(for)$$

$$\approx white + \text{noise}$$

HD Word Embeddings Summary

- Context embeddings: [dog] = A + white + went + for
- Order embeddings: <dog> = $\rho^{-2}(A) + \rho^{-1}(white) + \rho^{1}(went) + \rho^{2}(for)$
- How do we compute this though? What does “A” or “white” represent?

Label Vectors

- Label vectors: Immutable hypervectors that are used to represent words for the purposes of learning word embeddings
- We will denote with *word*
- Generated by randomly sampling elements to form a d-dimensional hypervector
  - Binary: {0, 1}
  - Ternary: {-1, 0, 1}
  - Bipolar: {-1, 1}
  - Others
- For our implementation, we will use sparse ternary hypervectors

dog = [-1 0 0 1 0 0 0 0 0 -1 0 -1 1 0 0 0 0 0 1]

Label Vectors

"A white dog went for a walk."

- <dog> = $\rho^{-2}(A) + \rho^{-1}(white) + \rho^{1}(went) + \rho^{2}(for)$

Order vector Label vector Label vector Label vector Label vector

- <dog> and [dog] are learned, *dog* is fixed

Label Vectors

We have label vectors, context vectors, order vectors. But what's the difference?

Label vectors	Context and Order vectors/embeddings
Fixed/Immutable	"Learned" over the course of training (mutable)
Initialized to have non-zero values	Initialized to be 0
Can be thought of as inputs	Can be thought of as outputs

Randomness

- Why do we want to randomly initialize the label vectors? Why not just set them to all be 0?
- How can we compare hypervectors?
  - Cosine distance:

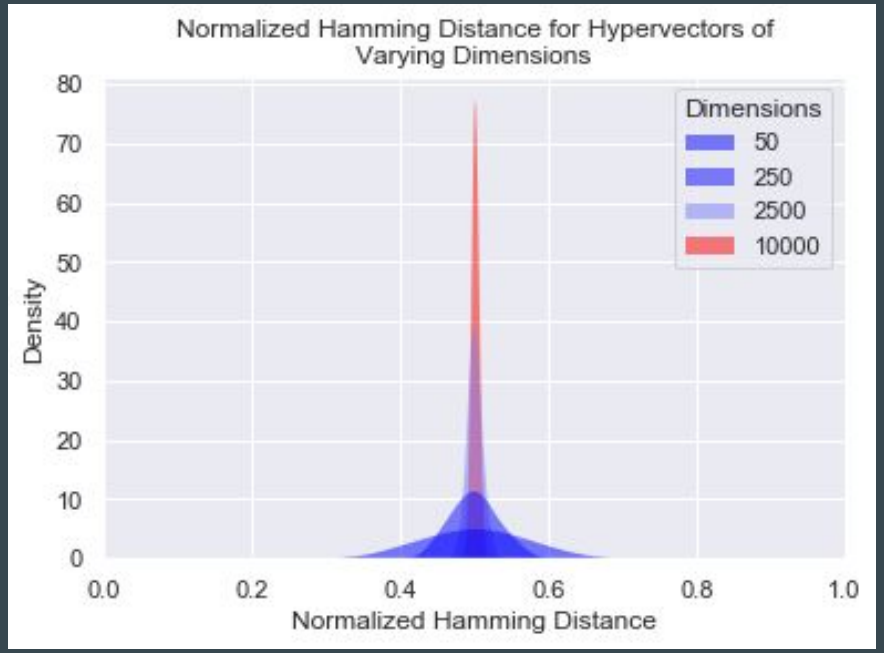
$$d_{\text{cosine}}(x, y) = \frac{x \cdot y}{|x|\,|y|}$$

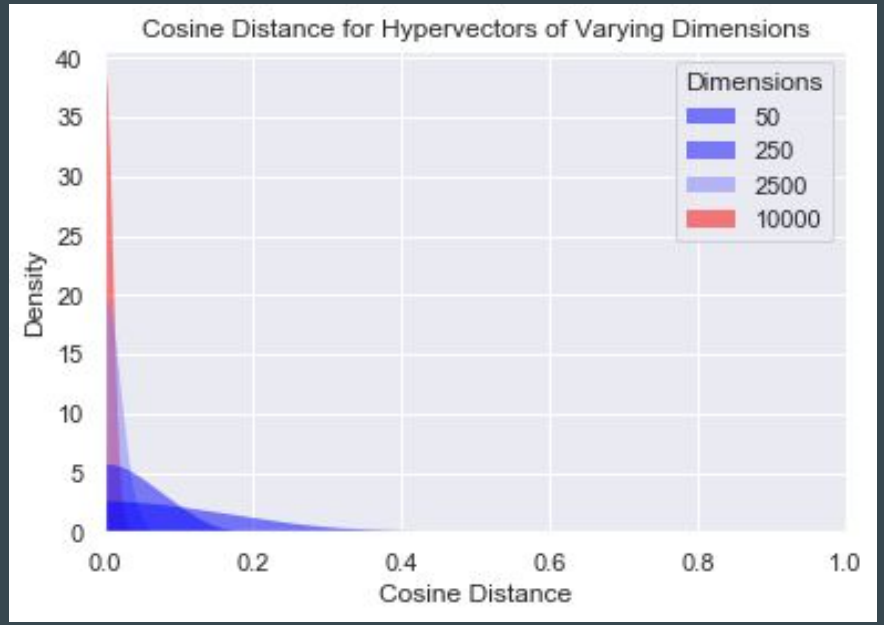
  - Hamming distance:

$$d_{\text{Hamming}}(x, y) = \sum_{i=1}^{d} 1\{x_i \neq y_i\}$$

- This is when the randomness and dimensionality comes into play

Randomness


Normalized Hamming Distance for Hypervectors of Varying Dimensions
Dimensions
50
250
2500
10000
Density
Normalized Hamming Distance
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80


Cosine Distance for Hypervectors of Varying Dimensions
Dimensions
50
250
2500
10000
Density
Cosine Distance
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
40

HD Word Embeddings Implementation

- We now know how to represent the words, how to learn the embeddings, and how to compare the embeddings that we learn
- So how do we put this all together?

HD Word Embeddings Implementation

Step 1: Define label vectors for each word in the vocabulary

Demo

HD Word Embeddings Implementation

Step 2: Iterate through each word in the corpus and compute the embedding

Demo

HD Word Embeddings Implementation

Step 3: Prosper

- So now we have these embeddings. What can we do with them?
- Some common uses:
  - TOEFL (Test of English as a Foreign Language)
    - Multiple choice
    - Good for semantic similarity
  - Word Clustering
    - Learn clusters of embeddings
    - Useful for POS and semantic tagging
  - Sentence Completion

Question	Option A	Option B	Option C	Option D
enormously	appropriately	uniquely	**tremendously**	decidedly
slowly	rarely	**gradually**	effectively	continuously
tranquillity	**peacefulness**	harshness	weariness	happiness
feasible	permitted	**possible**	equitable	evident
bigger	steadier	closer	**larger**	better

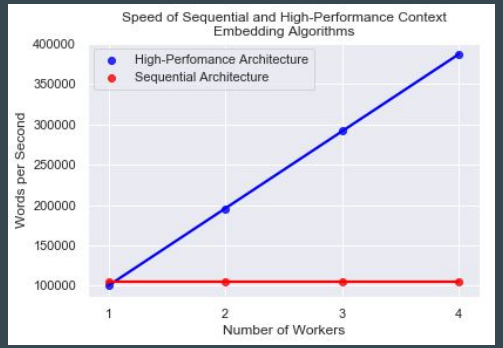
An Example of Changing Word Activation With Both Context and Order Information

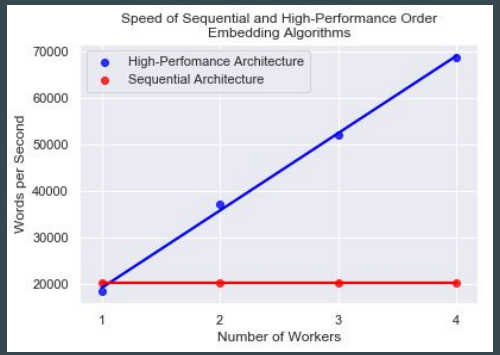
Probe	jefferson	edison	aquinas	paine	pickney	malthus
Thomas ____	**.72**	.66	.60	.35	.46	.34
Thomas ____ wrote the Declaration of Independence	**.44**	.30	.24	.29	.17	.14
Thomas ____ made the first phonograph	.33	**.45**	.29	.17	.21	.12
Thomas ____ taught that all civil authority comes from God	.30	.26	**.40**	.13	.17	.12
Thomas ____ is the author of *Common Sense*	.29	.21	.19	**.43**	.18	.13
A treaty was drawn up by the American diplomat Thomas ____	.32	.26	.27	.17	**.92**	.15
Thomas ____ wrote that the human population increased faster than the food supply	.23	.22	.21	.12	.14	**.41**

Jones, M. N., & Mewhort, D. J. (2007). Representing word meaning and order information in a composite holographic lexicon. Psychological review, 114(1), 1.

HD Word Embeddings Implementation

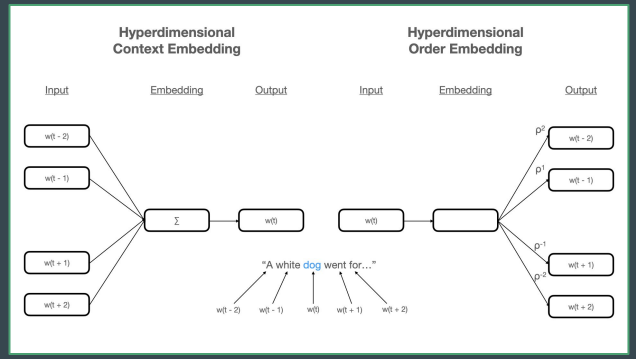
- This is impressively simple code to write
- Learning process is only a couple of lines in a while loop
- Computational time is linear with respect to the corpus size
- Easily parallelizable
- Memory is linear with respect to vocabulary size
- Empirically seems to learn quickly thanks to the nice statistical properties

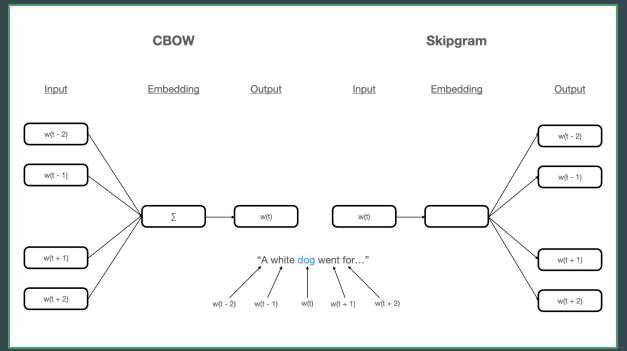




Similar Models

- How do these relate to other language models?
- Latent Semantic Analysis (LSA)
  - Very similar to context embeddings
  - Instead of label vectors, each word is one-hot encoded
    - Very space-inefficient
    - How much better is hd computing with regard to the memory?
- Word2Vec
  - Continuous Bag of Words (CBOW)
    - Same exact idea as context embeddings
    - Uses neural networks instead of hd computing
  - Skipgram
    - Similar idea to order embeddings
    - Uses neural networks instead of hd computing





My Disclaimer

- The distributional semantics hypothesis is a great way to define the problem in a way that makes sense to computers
- ... But I waver on how much faith I have in it

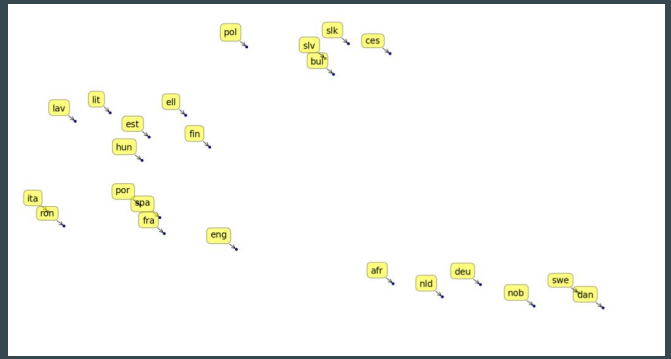
My Disclaimer

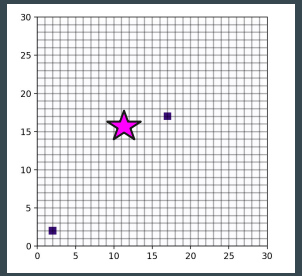
- Language is not just about characters on a page
- This leads me to be particularly interested in multimodal embeddings
  - Current neural network approaches seem to be rather large
  - Could HD computing-based embeddings simplify this?
    - Need only one pass via the training data

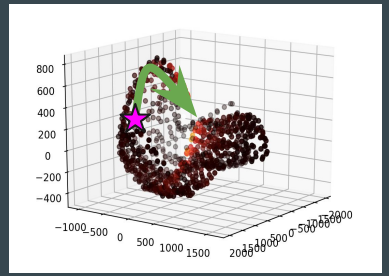
Applications

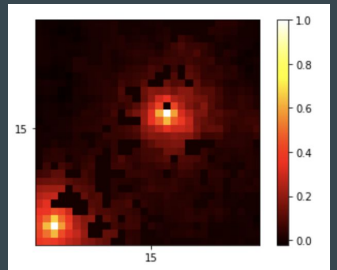
Applications

- We've seen how generating embeddings can be useful for language, but is there anything else?
- The short answer: yes!
  - Learning spatial manifolds
  - Language recognition
  - Some research I'm working on at LLNL









Questions?