Embedding Vectors

Jeff Allard

Lead Data Scientist

5/3 Bancorp

About

- Highly effective in several domains e.g. Natural Language Processing (Word2Vec)
- Finding more applications across machine learning problems generally

An Embedding Vector is simply a series of numbers that attempts to encode latent features of an object

- A word / sentence
- A customer
- A movie
- ……

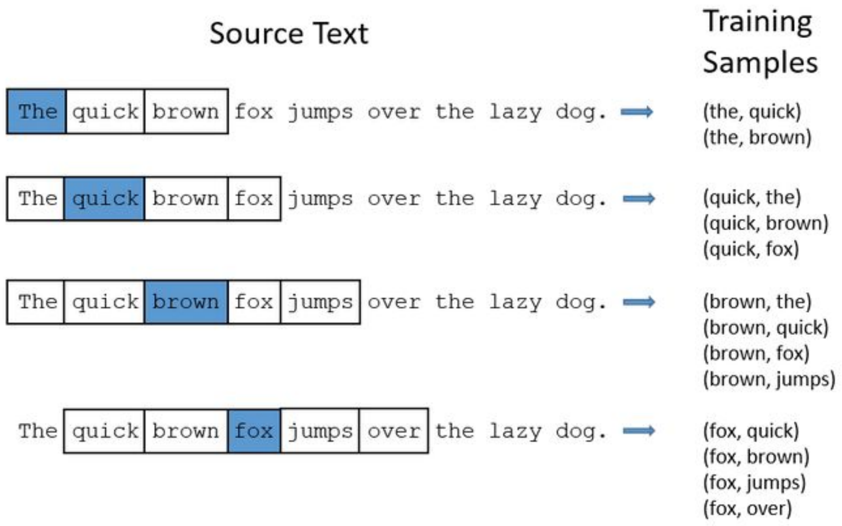
Canonical Example of Text Encoding

- Traditionally, models, say text classification, utilized a bag of words approach
  - “Cat”: [0,0,0…,0,0,1,0,0,….0] (length = size of vocabulary) where the vector is sparse - all zeros except for position “i” which represents “cat”
  - “Pet”: [0,0,1…,0,0,0,0,0,….0] (length = size of vocabulary) where the vector is sparse - all zeros except for position “j” which represents “pet” (i <>j)
- Lots of problems:
  - Very large sparse vectors are hard to work with in downstream algorithms
  - Related words encoded orthogonally
  - Need for heuristics such as word stemming, stop word removal, user maintained dictionaries of synonyms / domain specific words etc
- Variants such as TFIDF suffer same weakness

Solution: Word2Vec

- Originally published by Google researcher in 2013
- Many variants and subsequent alternatives
- The basic idea:
  - Represent a word as as dense vector
  - “Cat” = [0.2,-1.25,-0.0256,...0.45]
- Benefits
  - Size of the vector is much smaller than Bag-of-Words (e.g. 200)
  - Related words are similar in vector space terms (e.g cosine similarity)

Word2Vec : Basic Idea of Skip-gram method



- Window through a corpus
- Select a word (e.g. quick)
- Predict the likelihood that another word (e.g. fox) is within the context of the word -- context here is a window size of 2
- Train a neural network with positive and randomly chosen negative examples
- The weights from a shallow net are the vectors for a given word
- **Words with similar context will have learned representations that are similar**
  - E.g. Cat and Pet, Smart and Intelligent

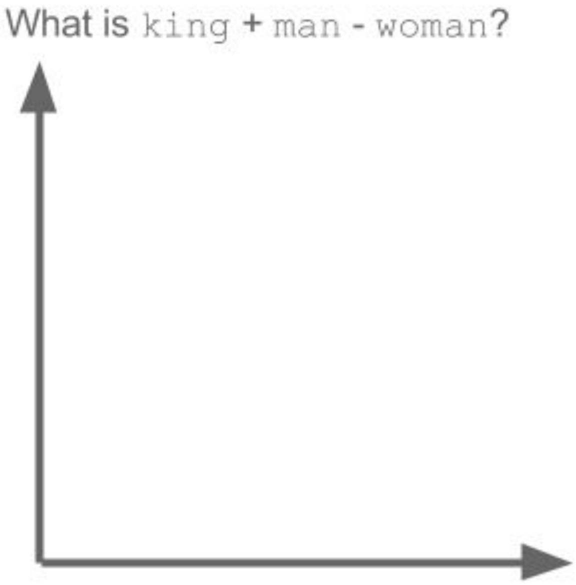
Word2Vec

- Intuitively....the elements of a vector describe concepts or characteristics of a term
  - E.g. One element of “King” may describe the strength of “gender”
  - Found that simple operations on these vectors displayed deeper concepts

[king] - [man] + [woman] ~= [queen]

- If we take the vector for “king”, subtract the vector for “man” and add the vector of “woman”, the resulting vector will be the closest to....the vector for queen (cosine similarity)
  - King - man removes the male gender component from the concept of monarch

Word2Vec



- [0.5,0.2] - [0.3,0.4] = [0.2,-0.2]
  (Man - Woman)
- [0.5,0.7] - [0.2,-0.2] = [0.3,0.9]
  (King + (Man - Woman)) ~ Queen

https://multithreaded.stitchfix.com/assets/images/blog/vectors.gif

Word2Vec

- Closing Thoughts
  - Intuitively the components of the vectors are concepts or mixtures of concepts
  - In practice not always interpretable
  - The vector dimension is often chosen empirically
  - Easy to train vectors on your own domain if you have enough data, else use pre-trained ones (e.g. trained on massive corpus like wikipedia) and fine tune them (transfer learning) for your task

Entity Embeddings

- Generalization - Not just for words or text
- A word is just a some token, so extend this idea to things like
  - Customer IDs
  - Movie IDs
  - Zip codes
  - ......
- Being used for state of the art predictive models
  - Allows use of high cardinality nominal variables like a customers census tract
  - Even used for lower cardinality variables like a customers State (replacing dummy variables)

Application : Recommendation Engines

User Latent Factors			
#1	#2	#3	User ID
0.8949101073	-0.6265794545	0.74757374	0
0.7947220104	0.8636220878	0.312756333	1
-0.2558703818	0.8342031778	0.081368916	2
0.940665854	-0.5790454715	0.962063664	3
0.9222317515	0.9514060606	0.432131189	4
0.8462968955	0.1530153498	-0.50570987	5
0.1811727399	0.1472382755	0.756643752	6

Movie Latent Factors			
#1	#2	#3	Movie ID
0.8239053285	0.8610092645	0.083420775	0
0.3149946319	0.2860168437	0.179063335	1
0.528859128	-0.8185830713	-0.519957348	2
-0.3441695914	0.1525521576	0.133796819	3
0.851516663	-0.2228508417	0.227523797	4

User Factor #1: How much the user likes action movies?

Movie Factor #3: How much action the movie has?

Demo

- Highly effective deep learning model for recommending items to Pintrest users
- Using only user and item IDs
- Training from scratch with gradient descent
- Keras and Tensorflow

GITHUB ADDRESS