Table of contents

1. Intro
2. Word Vectors
3. Word2Vec
4. Char Level Word Embeddings
5. Application: Entity Matching
6. Conclusion
Intro
Intro

Deep Learning
Core Idea

• Given data that has hard to describe complex intrinsic structure
  • Important: DL usually works well only when data has unknown hidden structure. Not a silver bullet for all tasks
• Learn hierarchical representations of data points to perform a task
• Input representation
• Hidden representations (higher level features)
• Output prediction
Main Challenge

• Assisting the neural network to learn to create good representations
• How?
  • Encoding domain knowledge
  • Make training easier
    • Improved optimization techniques
    • Better gradient flow
Model Engineering

- DL represents a shift in the way we think about ML
  - Traditional ML: Encode domain knowledge using feature engineering
  - “Deep” ML: Encode domain knowledge using model engineering
Intro

Natural Language Processing
What is NLP? (from Stanford CS224n)

- **Natural language processing** is a field at the intersection of:
  - computer science
  - artificial intelligence
  - and linguistics.
- **Goal**: for computers to process or “understand” natural language in order to perform tasks that are useful, e.g.,
  - Performing Tasks, like making appointments, buying things
  - Question Answering
    - Siri, Google Assistant, Facebook M, Cortana ... thank you, mobile!!!
- Fully **understanding and representing** the **meaning** of language (or even defining it) is a difficult goal.
  - Perfect language understanding is AI-complete
Applications (from Stanford CS224n)

- Search (written and spoken)
- Online advertisement matching
- Automated/assisted translation
- Sentiment analysis for marketing or finance/trading
- Speech recognition
- Chatbots / Dialog agents
  - Automating customer support
  - Controlling devices
  - Ordering goods
The NLP Formula: Hierarchical Feature Generation

- Input representation (Word Vectors)
- Hidden representations (Typically using LSTM RNNs)
- Output prediction (Typically using a variant of softmax)
Word Vectors
Word Vectors

Motivation
What do we want?

We want a representation of words that:

• Captures meaning
• Is efficient
Past attempts to capture meaning of words - WordNet

```python
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

(here, for **good**):

- S: (adj) full, good
- S: (adj) estimable, good, honorable, respectable
- S: (adj) beneficial, good
- S: (adj) good, just, upright
- S: (adj) adept, expert, good, practiced, proficient, skillful
- S: (adj) dear, good, near
- S: (adj) good, right, ripe
- ...
- S: (adv) well, good
- S: (adv) thoroughly, soundly, good
- S: (n) good, goodness
- S: (n) commodity, trade good, good
A simple idea

“You shall know a word by the company it keeps” (J. R. Firth 1957: 11)
Ways to create word embeddings

• Traditional count based methods
  • SVD, PPMI, etc.

• Neural network based methods
  • Word2Vec et al.

• Hybrid
  • Glove
Word Vectors

Language Modeling
What is a Language Model?

- A probability distribution over sequences of words
  \[ P(w_1, \cdots, w_m) = \prod_{t=1}^{m} P(w_t|w_{t-1}, \cdots, w_1) \]
- In practice, we only condition on the last \( M \) words
  \[ P(w_1, \cdots, w_m) = \prod_{t=1}^{m} P(w_t|w_{t-1}, \cdots, w_{t-M}) \]
Training word vectors using LMs (Bengio et al. 2003)
Word2Vec
Core Idea

- Parameterize words as vectors
- Use an NN to *predict* nearby words
  - Skip Gram
  - CBOW
- Hope that learned word vectors are semantically meaningful
Skip Gram Model
For each word $t = 1 \ldots T$, predict surrounding words in a window of “radius” $m$ of every word.

Objective function: Maximize the probability of any context word given the current center word:

$$
J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{m \leq j \leq m \\ j \neq 0}} p(w_{t+j} | w_t; \theta)
$$

Negative Log Likelihood

$$
J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{m \leq j \leq m \\ j \neq 0}} \log p(w_{t+j} | w_t)
$$

Where $\theta$ represents all variables we will optimize.
Predict surrounding words in a window of radius $m$ of every word

For $p(w_{t+j} | w_t)$ the simplest first formulation is

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

where $o$ is the outside (or output) word index, $c$ is the center word index, $v_c$ and $u_o$ are “center” and “outside” vectors of indices $c$ and $o$

**Softmax** using word $c$ to obtain probability of word $o$
Step by step (from Stanford CS224n)

Skipgram

\[ V \times 1 \quad d \times V \quad d \times 1 \]

\[ W = \text{softmax}(u^T V) \]

\[ p(x|c) = \text{softmax}(u^T V) \]

\[ \text{Softmax} \]

\[ p_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \]

Actual context words

Output word representation

Looks up column of word embedding matrix as representation of center word

One hot word symbol
The problem of Softmax

- Scalability wrt vocabulary size
  - Do we really need to compute the probability of ALL words during training?
  - Do we really need to push down the prob. of ALL wrong words during training?

- Solutions:
  - Hierarchical Softmax
  - Negative sampling
The problem of Softmax

Skipgram

$V \times 1 \quad d \times V \quad d \times 1$

$W = Ww_c$

$W' = [Ww_c]$ 

$V_{x \times d}$

$P(x|c) = \text{softmax}(u_x^T V_c)$

$V_{x \times 1}$

$V_{t-2}$

$V_{t-1}$

$V_{t+1}$

$W_{t-3}$

Softmax $\pi_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$

Actual context words

Output word representation

Looks up column of word embedding matrix as representation of center word
• “Preprocess” step:
  • Randomly split the vocabulary $V$ into $\sqrt{|V|}$ clusters consisting of approximately $\sqrt{|V|}$ words each
  • I.e., each word $w$ in $V$ is permanently assigned to a cluster $C(w)$ at the beginning of training
Hierarchical Softmax - Simple version

The Idea

- Compute the probability $p(w_{t-1}|\vec{c}_t)$ in three steps:
  - First, compute the probability of the word $w_{t-1}$ belonging to its true cluster $C(w_{t-1})$, i.e., $p(C(w_{t-1})|\vec{c}_t)$
  - Next, compute the probability of the word within its true cluster $C(w_{t-1})$, $p(w_{t-1}|C(w_{t-1}), \vec{c}_t)$
  - Then, $p(w_{t-1}|\vec{c}_t)$ is the product of the two probabilities:

  $$p(w_{t-1}|\vec{c}_t) = p(w_{t-1}, C(w_{t-1})|\vec{c}_t) = p(C(w_{t-1})|\vec{c}_t) \times p(w_{t-1}|C(w_{t-1}), \vec{c}_t)$$
Hierarchical Softmax - Simple version

- Computing cluster probability $p(C(w_{t-1})|\vec{c}_t)$:
  - Use a regular softmax NN with $\sqrt{|V|}$ classes, 1 for each cluster.
Hierarchical Softmax - Simple version

- Computing word in cluster probability $p(w_{t-1}|C(w_{t-1}), \tilde{c}_t)$:
  - To do so, each word cluster has a dedicated softmax NN with approximately $\sqrt{|V|}$ classes, 1 for each word in the cluster.
Hierarchical Softmax

- Time complexity of simple version = $O(\sqrt{|V|})$
- Can do much better ($O(\log |V|)$) with more complex versions
  - Frequency based clustering of words
  - Brown clustering
  - etc...
Negative Sampling

- Radical alternative
- Instead of literally predicting nearby words, we learn to distinguish true nearby words from randomly selected words.
- Consider example of predicting $w_{t-1}$
  - Sample $k$ negative words $n_{t-1}^1, \cdots, n_{t-1}^k \in V$, i.e., $k$ words in $V$ other than $w_{t-1}$
  - Compute dot products only for the target word $w_{t-1}$ and the $k$ negative samples.
  - Instead of computing a probability using a softmax, the model is supposed to make a binary decision for each word using a sigmoid, i.e., predict 1 for word $w_{t-1}$ and 0 for words $n_{t-1}^1, \cdots, n_{t-1}^k$. 
<table>
<thead>
<tr>
<th>Relationship</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>France - Paris</td>
<td>Italy: Rome</td>
<td>Japan: Tokyo</td>
<td>Florida: Tallahassee</td>
</tr>
<tr>
<td>big - bigger</td>
<td>small: larger</td>
<td>cold: colder</td>
<td>quick: quicker</td>
</tr>
<tr>
<td>Miami - Florida</td>
<td>Baltimore: Maryland</td>
<td>Dallas: Texas</td>
<td>Kona: Hawaii</td>
</tr>
<tr>
<td>Einstein - scientist</td>
<td>Messi: midfielder</td>
<td>Mozart: violinist</td>
<td>Picasso: painter</td>
</tr>
<tr>
<td>Sarkozy - France</td>
<td>Berlusconi: Italy</td>
<td>Merkel: Germany</td>
<td>Koizumi: Japan</td>
</tr>
<tr>
<td>copper - Cu</td>
<td>zinc: Zn</td>
<td>gold: Au</td>
<td>uranium: plutonium</td>
</tr>
<tr>
<td>Berlusconi - Silvio</td>
<td>Sarkozy: Nicolas</td>
<td>Putin: Medvedev</td>
<td>Obama: Barack</td>
</tr>
<tr>
<td>Microsoft - Windows</td>
<td>Google: Android</td>
<td>IBM: Linux</td>
<td>Apple: iPhone</td>
</tr>
<tr>
<td>Microsoft - Ballmer</td>
<td>Google: Yahoo</td>
<td>IBM: McNealy</td>
<td>Apple: Jobs</td>
</tr>
<tr>
<td>Japan - sushi</td>
<td>Germany: bratwurst</td>
<td>France: tapas</td>
<td>USA: pizza</td>
</tr>
</tbody>
</table>

E.g.: France - Paris + Italy ≈ Rome
• Learns word vectors independently for each word
• Cannot estimate word embeddings of out-of-vocabulary (OOV) words
Char Level Word Embeddings
Core Idea

- Make use of fact that words are composed of characters
- Parameterize characters as vectors
- Compose char vectors to create word vectors
  - Can use CNN or RNN
- Use an NN to predict nearby words
- Hope that word vectors are semantically meaningful
The Word Embedding Network (WEN)
Step 1: Lookup Char Embeddings

- Same as word vector lookup layer
Step 2: Temporal Convolution over Characters

• Assumption: Words are made of smaller units of character n-grams (morphemes)
• How do we encode this knowledge?
  • Use $k$ Temporal CNNs with $k$ kernel widths
  • Intuition: A CNN with kernel width $j$ detects the presence of character n-grams of length $j$
Step 2: Temporal Convolution over Characters

- Example: Temporal convolution with kernel width = 3 and two feature maps sensitive to 3-grams “mac” and “est”

<table>
<thead>
<tr>
<th>Character</th>
<th>Activations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mac</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
</tr>
</tbody>
</table>

Example:
- Kernel width = 3
- Feature maps: “mac” and “est”
Step 2: Temporal Convolution over Characters

- The output of step 2 consists of two pieces of information:
  - Which character n-grams are present in a given input word?
  - Where do they occur in the word?
Step 3: Max-Pooling

• For each feature map in each CNN, output only the maximum activation
• Intuition: Get rid of the position information of character n-grams
• This yields a fixed dimensional vector
Step 4: Fully Connected Layers

• 3-layer “highway network”
• Intuition: Compose a word vector using information about which character n-grams are present in the word
Step 4: Highway NN Layer Details

- Core Idea: Provide a means to allow inputs to flow unaltered through an NN layer
  - This improves gradient flow
- Alternate pathways (highways) conditionally allows the input to pass through undeterred to the corresponding outputs as if the layer did not exist
- “Gates” determine whether to modify each dimension of the input
Given an input vector $\vec{q}$, the output of the $n^{th}$ highway layer with input $\vec{x}_n$ and output $\vec{y}_n$ is computed as follows:

$$\vec{x}_n = \begin{cases} \vec{q}, & \text{if } n = 1, \\ \vec{y}_{n-1}, & \text{if } n > 1 \end{cases}$$

$$H(\vec{x}_n) = \vec{W}_n^{(H)} \vec{x}_n + \vec{b}_n^{(H)}$$

$$T(\vec{x}_n) = \vec{W}_n^{(T)} \vec{x}_n + \vec{b}_n^{(T)}$$

$$\vec{y}_n = H(\vec{x}_n) \odot T(\vec{x}_n) + \vec{x}_n \odot (1 - T(\vec{x}_n))$$

Here $H$ applies an affine transformation to $\vec{x}_n$ and $T$ acts as the gate deciding which inputs should be transformed and by how much.
The Word Embedding Network (WEN)
• How can we train this model to produce good word embeddings?
  • Train it as part of a recurrent neural language model
Training

Diagram showing the training process with layers labeled HP, LSTM, HSM, and output probabilities P(laptop), P(new), P(EOS).
<table>
<thead>
<tr>
<th>In Vocabulary</th>
<th>Out-of-Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>while</td>
<td>computer-aided</td>
</tr>
<tr>
<td>although</td>
<td></td>
</tr>
<tr>
<td>letting</td>
<td></td>
</tr>
<tr>
<td>though</td>
<td></td>
</tr>
<tr>
<td>minute</td>
<td></td>
</tr>
<tr>
<td>his</td>
<td>computerized</td>
</tr>
<tr>
<td>your</td>
<td>informed</td>
</tr>
<tr>
<td>conservatives</td>
<td>look</td>
</tr>
<tr>
<td>you</td>
<td></td>
</tr>
<tr>
<td>we</td>
<td></td>
</tr>
<tr>
<td>jonathan</td>
<td></td>
</tr>
<tr>
<td>rich</td>
<td></td>
</tr>
<tr>
<td>trading</td>
<td></td>
</tr>
<tr>
<td>advertised</td>
<td></td>
</tr>
<tr>
<td>advertising</td>
<td></td>
</tr>
<tr>
<td>turnover</td>
<td></td>
</tr>
<tr>
<td>Nancy</td>
<td></td>
</tr>
<tr>
<td>LSTM-Word</td>
<td></td>
</tr>
<tr>
<td>LSTM-Char</td>
<td></td>
</tr>
<tr>
<td>(before highway)</td>
<td></td>
</tr>
<tr>
<td>meanwhile</td>
<td>computer-guided</td>
</tr>
<tr>
<td>whole</td>
<td>performed</td>
</tr>
<tr>
<td>white</td>
<td>look</td>
</tr>
<tr>
<td>is</td>
<td></td>
</tr>
<tr>
<td>four</td>
<td></td>
</tr>
<tr>
<td>youth</td>
<td></td>
</tr>
<tr>
<td>Richter</td>
<td></td>
</tr>
<tr>
<td>heading</td>
<td></td>
</tr>
<tr>
<td>training</td>
<td></td>
</tr>
<tr>
<td>hard</td>
<td></td>
</tr>
<tr>
<td>rich</td>
<td></td>
</tr>
<tr>
<td>reading</td>
<td></td>
</tr>
<tr>
<td>richer</td>
<td></td>
</tr>
<tr>
<td>leading</td>
<td></td>
</tr>
<tr>
<td>computer</td>
<td></td>
</tr>
<tr>
<td>disk-drive</td>
<td></td>
</tr>
<tr>
<td>transformed</td>
<td></td>
</tr>
<tr>
<td>inform</td>
<td></td>
</tr>
<tr>
<td>looks</td>
<td></td>
</tr>
<tr>
<td>shook</td>
<td></td>
</tr>
<tr>
<td>LSTM-Char</td>
<td></td>
</tr>
<tr>
<td>(after highway)</td>
<td></td>
</tr>
<tr>
<td>meanwhile</td>
<td>computer-guided</td>
</tr>
<tr>
<td>whole</td>
<td>performed</td>
</tr>
<tr>
<td>though</td>
<td>look</td>
</tr>
<tr>
<td>nevertheless</td>
<td></td>
</tr>
<tr>
<td>nevertheless</td>
<td></td>
</tr>
<tr>
<td>meanwhile</td>
<td></td>
</tr>
<tr>
<td>whole</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td></td>
</tr>
<tr>
<td>your</td>
<td></td>
</tr>
<tr>
<td>we</td>
<td></td>
</tr>
<tr>
<td>Gerard</td>
<td></td>
</tr>
<tr>
<td>trading</td>
<td></td>
</tr>
<tr>
<td>traded</td>
<td></td>
</tr>
<tr>
<td>computer</td>
<td></td>
</tr>
<tr>
<td>computer-driven</td>
<td></td>
</tr>
<tr>
<td>transformed</td>
<td></td>
</tr>
<tr>
<td>outperformed</td>
<td></td>
</tr>
<tr>
<td>looked</td>
<td></td>
</tr>
<tr>
<td>looking</td>
<td></td>
</tr>
</tbody>
</table>
Application: Entity Matching
The Task

- Given text descriptions of two entities, determine if they refer to the same real world entity
- Binary classification problem

(a) LG 32MA68HY-P 32-Inch IPS Monitor. The 32-inch Class screen provides 75% more viewable area compared to a 24-Inch monitor. The 32-inch Class screen (31.5” measured diagonally) provides 75% more Viewable area compared to a 24-Inch monitor. The full HD clarity and true Color reproduction of the 16:9 IPS panel enhance every project, even when viewed off-angle. […]

(b) The 48in class 32MA68HY-P 18:10 monitor from LG offers more than a 60% larger screen area vs. a 32in class monitor. That's plenty of room for work on spreadsheets, editing documents, watching videos, viewing photos and more. Supports Display Port, HDMI, D-Sub, USB 2.0 (x2), and USB 3.0. […]

45
The Task

- Typical approach: Information Extraction (IE)
  - Followed by training a traditional classifier: SVM, Random Forest etc.
- Proposed alternate approach: Neural Entity Matching Model (Nemo)
  - Use deep learning to perform this end to end
  - Train a single model to perform extraction and entity matching
Nemo Overview

Diagram showing the flow of data from sequences of words to word embeddings, then through attribute and entity embeddings, to entity comparison network leading to a prediction.
Baseline - Magellan

- Generates string similarity metrics based on heuristics
  - Jaccard, Cosine, Levenshtein etc.
- Picks the best standard ML model using cross validation
  - SVM, Random forest etc.
Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nemo</th>
<th>Magellan</th>
<th>$F_1$ absolute improvement</th>
<th>$F_1$ relative improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P$</td>
<td>$R$</td>
<td>$F_1$</td>
<td></td>
</tr>
<tr>
<td>Home</td>
<td>89.2</td>
<td>90.0</td>
<td><strong>89.6</strong></td>
<td>10.6</td>
</tr>
<tr>
<td>Electronics</td>
<td>91.2</td>
<td>93.7</td>
<td><strong>92.4</strong></td>
<td>9.5</td>
</tr>
<tr>
<td>Tools</td>
<td>94.5</td>
<td>95.9</td>
<td><strong>95.2</strong></td>
<td>7.2</td>
</tr>
<tr>
<td>Clothing</td>
<td>96.4</td>
<td>97.5</td>
<td><strong>97.0</strong></td>
<td>13.9</td>
</tr>
<tr>
<td>Company</td>
<td>90.1</td>
<td>87.8</td>
<td><strong>89.0</strong></td>
<td>4.3</td>
</tr>
</tbody>
</table>
## Results - Extensive extraction

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nemo</th>
<th>Extracting top-5 attributes</th>
<th>Magellan</th>
<th>Extracting top-30 attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P$</td>
<td>$R$</td>
<td>$F_1$</td>
<td>$P$</td>
</tr>
<tr>
<td>Home</td>
<td>89.2</td>
<td>90.0</td>
<td>89.6</td>
<td>90.1</td>
</tr>
<tr>
<td>Electronics</td>
<td>91.2</td>
<td>93.7</td>
<td>92.4</td>
<td>94.8</td>
</tr>
<tr>
<td>Tools</td>
<td>94.5</td>
<td>95.9</td>
<td>95.2</td>
<td>95.9</td>
</tr>
<tr>
<td>Clothing</td>
<td>96.4</td>
<td>97.5</td>
<td>97.0</td>
<td>97.5</td>
</tr>
</tbody>
</table>
Understanding the model

• Compute first order saliency scores for word embeddings
  • Essentially the derivative of word embeddings wrt the output of the model
  • Measure of sensitivity
Example:

- Assume we have an entity pair \((e_1, e_2)\)
- We want to compute the saliency of the \(t^{th}\) word in \(e_1\)
- If \(y_L\) represents the score that Nemo assigns to the correct label \(L\) (match or mismatch) for the pair, \(\vec{x}_t\) represents Nemo’s input units for the \(t^{th}\) word and \(\vec{w}_t\) is the word embedding of the \(t^{th}\) word in \(e_1\), then the saliency \(s\) of the word is calculated as

\[
s = \left\| \frac{\partial y_L}{\partial \vec{x}_t} \bigg|_{\vec{x}_t=\vec{w}_t} \right\|
\]
<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Home</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>any</td>
<td>manufacturing</td>
<td>defects</td>
<td>li</td>
<td>colonial</td>
<td>mills</td>
<td>twilight</td>
<td>palm</td>
<td>area</td>
<td>rug</td>
<td>home</td>
<td>&amp;</td>
<td>garden</td>
<td>features</td>
<td>technique</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>silver</td>
<td>metal</td>
<td>armless</td>
<td>ships</td>
<td>individually</td>
<td>tempo</td>
<td>vinyl</td>
<td>tlp1349</td>
<td>br</td>
<td>features</td>
<td>ul</td>
<td>li</td>
<td>stationary</td>
<td>stool</td>
<td>li</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>height</td>
<td>top</td>
<td>to</td>
<td>bottom</td>
<td>16</td>
<td>inches</td>
<td>li</td>
<td>li</td>
<td>overall</td>
<td>width</td>
<td>side</td>
<td>to</td>
<td>side</td>
<td>16</td>
<td>inches</td>
<td></td>
</tr>
</tbody>
</table>

| **Company** |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| (d) | group | started | about | 70 | years | ago | at | kyoto | uzumasa | japan | the | center | of | the | japanese |     |     |
| (e) | the | deltic | group | exciting | and | entertaining | venues | the | deltic | group | exciting | and | entertaining | venues | menu |     |     |
| (f) | b. | riley | financial | on | july | 1 | 2016 | robert | j. | taragan | became | the | new | ceo | of |     |     |
Entity Embedding Visualization
Entity Embedding Visualization - Cables

- Smart Buy - 19 RIGHT

- 100FT FLEXboot Series 24AWG Cat6 550MHz UTP Bare C RIGHT

- GENERAL CABLE CD932A-41.10 Computer Cable 24 AWG C RIGHT

- Comprehensive Cable USB3-AA-65T USB 3.0 A Male To RIGHT

- Tripp Lite Gold w/RGB Coax - VGA cable - HD-15 [M] RIGHT

- Tripp Lite N082-014-BL Molded Cable - 14-ft Cat5e RIGHT

- Cables To Go 10-Foot DVI-D Dual Link Male/Male Cab RIGHT

- Qgear Mini Displayport To Displayport Cable - 6F RIGHT

- TP-LINK TL-ANT24EC6N Low-loss Antenna Cable - 6 Me RIGHT

- Insten 25 USB 2.0 A to A Male to Female Extension RIGHT

- DisplayPort (M) to HDMI (F) Adapters RIGHT

- OFCON-1000B 1000ft Cat5e Blue Plenum RIGHT

- 625 MHz AVD-1000B 1000ft Cat5e Blue Plenum RIGHT
Entity Embedding Visualization - Bags

- David King & Co 3522G Florentine Flap Front Handbag RIGHT
- SW Global Carina Side Studded Satchel LEFT
- Magid Lurex Stripe Paper Tote LEFT
- 18 Robert McClintock Beagle Painting Tote Bag RIGHT
- Littlearth Hoodie Purse - NFL Teams LEFT
- Reusable Cotton Tote Bag w/Zipper - Natural/Blue Tr RIGHT
- Le Donne Leather Hobo With Side Zip Pockets LEFT
- David King & Co. Florentine Flap Front Handbag LEFT
Entity Embedding Visualization - Desks
Conclusion
Deep Learning: An incredibly promising tool

- Has produced breakthroughs in several domains
- But has a lot of open problems
  - Large quantities of labeled data
  - Huge number of parameters
  - Optimization challenges