Table of Content

1. Motivation
2. Background:
   - F.Hill et al. Paper
   - J.Chen et al. Paper
3. Models:
   - LSTM
   - CNN (K.Yoon Paper)
4. Training & Results
5. Evaluation
   - T.Schnabel et al. Paper
   - Tensor2Tensor

Part 2. Improvement Idea
1. Closer look at word embedding algorithms
2. Problem with current method
3. Atoms of discourse
4. Solution
   - S. Arora et al. Paper
5. Results

Part 3. Discussion
1. Possible Applications
2. Related Works
   - Sememes
   - Jump LSTM
   - Elastic Weight Consolidation
Word Embeddings & Dictionaries
Motivation: Word Vector Problems

- Rare words (e.g. min_count = n)
- Opposite words (e.g. good and bad)
- Words with multiple meaning (e.g. bat, ring, bush)
During reading, when we do not understand a word, we look it up in the dictionary.
Understand Phrases by Embedding the Dictionary

- Idea: Train a Seq2Vec model using a word’s dictionary definition as input and it’s word vector as label
- Application: reverse dictionary and crossword puzzles
- Yoshua Bengio & TACL(2016)
<table>
<thead>
<tr>
<th>Test Set</th>
<th>Dictionary definitions</th>
<th>Concept descriptions (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seen (500 WN defs)</td>
<td>Unseen (500 WN defs)</td>
</tr>
<tr>
<td></td>
<td>median rank</td>
<td>accuracy@10/100</td>
</tr>
<tr>
<td>Unsup. models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2V add</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>W2V mult</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OneLook</td>
<td>0 .89/91</td>
<td>67</td>
</tr>
<tr>
<td>NLMs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN cosine</td>
<td>12 .48/.73</td>
<td>103</td>
</tr>
<tr>
<td>RNN w2v cosine</td>
<td>19 .44/.70</td>
<td>111</td>
</tr>
<tr>
<td>RNN ranking</td>
<td>18 .45/.67</td>
<td>128</td>
</tr>
<tr>
<td>RNN w2v ranking</td>
<td>54 .32/.56</td>
<td>155</td>
</tr>
<tr>
<td>BOW cosine</td>
<td>22 .44/.65</td>
<td>129</td>
</tr>
<tr>
<td>BOW w2v cosine</td>
<td>15 .46/.71</td>
<td>124</td>
</tr>
<tr>
<td>BOW ranking</td>
<td>17 .45/.68</td>
<td>115</td>
</tr>
<tr>
<td>BOW w2v rankng</td>
<td>55 .32/.56</td>
<td>155</td>
</tr>
<tr>
<td>Input Description</td>
<td>OneLook</td>
<td>W2V add</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
</tbody>
</table>
Learning Word Embeddings from Intrinsic and Extrinsic Views

- Extrinsic: Context Information
- Intrinsic: Definitions & Explanations
Learning Word Embeddings from Intrinsic and Extrinsic Views

\[
L = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log \sigma(v_{wt}^T v'_{wt+j}) + \sum_{i=1}^{k} E_{wi} \sim P_n(w) \left[ \log \sigma(-v_{wt}^T v'_{wi}) \right],
\]

VS

\[
L = \sum_{-c \leq j \leq c, j \neq 0} \log \sigma(v_{wt}^T v'_{wt+j}) + \sum_{i=1}^{k} E_{wi} \sim P_n(w) \left[ \log \sigma(-v_{wt}^T v'_{wi}) \right] + \log \sigma(v_{wt}^T v_{R(wt)})
\]
## Quantitative Result

<table>
<thead>
<tr>
<th>Method</th>
<th>WS-353</th>
<th>MEN</th>
<th>MTurk-771</th>
<th>YP-130</th>
<th>SimLex-999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>44.57%</td>
<td>37.08%</td>
<td>31.95%</td>
<td>4.25%</td>
<td>17.88%</td>
</tr>
<tr>
<td>Glove</td>
<td>45.35%</td>
<td>32.93%</td>
<td>35.29%</td>
<td>8.64%</td>
<td>20.04%</td>
</tr>
<tr>
<td>DEWE</td>
<td>43.97%</td>
<td>38.47%</td>
<td>36.93%</td>
<td>13.82%</td>
<td>21.46%</td>
</tr>
</tbody>
</table>
Def2Vec Models

- CNN (Kim Yoon, 2014)
- LSTM (Original Paper)
- Based on CNN from Yoon Kim (2014) paper on sentence classification
- Single convolutional layer with no pooling, and single fully connected layer.
- Zero padding to make definitions same length.
- Dropout after fully connected.
- Implemented with tf.contrib.rnn
- Final hidden state of LSTM as vector representation of the definition.
- Original paper have a linear mapping module at the end to convert internal state dimension to word vector dimension. Omitted here because, the internal state have the same dimension as word vector
Data

- Dictionary definition from multiple dictionaries (mainly WordNet) using Wordnik.com API

- Approximately 200,000 definitions of 75,000 words (original paper also trained on pseudo-definition from wikipedia, so it used Approximately 900,000 definition of 100,000 words)

- All word converted to 500D word vectors using pre-trained Glove word embedding (Wikipedia + Gigaword 5). Unknown words randomly initialized.
Def2Vec Training Parameters

**CNN**
- **Input**: n*500 matrix
- **Window Size**: 3,4,5
- **Feature Map Size**: 500
- **Fully Connected Size**: 1024
- **Dropout Rate**: 0.5
- **Batch Size**: 50
- `tf.train.AdadeltaOptimizer`

**LSTM**
- **Input**: sequence of 500D vectors
- **Internal State Size**: 500
- **Batch Size**: 16
- `tf.train.AdadeltaOptimizer`
\[
\max(0, m - \cos(M(s_c), v_c) - \cos(M(s_c), v_r))
\]
- 500 randomly picked Wordnet definitions
- Top 10 closest word vector according to cosine distance
37.1%
CNN accuracy @ top 10

39.8%
LSTM accuracy @ top 10
- Similar but worse than original result as expected, because of less training data.
- Overall, very poor accuracy, but prediction roughly in the region of correct word vector.
- Something is fundamentally wrong about this method.
Evaluation Methods for Unsupervised Word Embeddings

- Comprehensive study of evaluation methods
- Linguistic Insight:
  - Relatedness (similar = nearby?)
  - Coherence (group = related?)
- Downstream Tasks:
  - Sentiment Classification
  - Machine Translation
Tensor2Tensor

- Very new TensorFlow package that has a very modular architecture.
- State of the art models, datasets, and hyperparameters available
- Very efficient training on those models
- Problem: LSTM related model not working
- Trained Google’s Transformer model on NMT task.
  - 70% Accuracy, 90% Top5 on WMT_ENDE_8k data
Improvement Idea
Any ideas?
Closer Look at Word Embedding Algorithms

- Context based
- Implicitly or explicitly use co-occurrence matrix
- One vector per word
Analysis & Problem

- Based on our data, one word have more than 2 definitions on average
- Context-based methods do not distinguish different meaning
- Definition must be precise, but the word vector might not be

**Problem: Words with multiple word senses**
- WordNet: large lexical database of English
  - Grouped words that links to other groups
  - Human generate word sense: short definitions
- S.Arora Paper: Linear Algebra Structure of Word Senses, with Applications to Polysemy
Directions in word embedding represent topic or discourse

Compare unit vector of topic to get their similarity
  ▶ Unit vector because we use dot product to compute similarity
  ▶ Look similar to human if dot product > 0.85, different if < 0.5

The whole embedding is comprised of m topics
Thought Experiment

\[ v_{w_{new}} \approx \alpha v_{w_1} + \beta v_{w_2} \]

\[ \beta \approx 1 - c \log r \]
Given word vectors in $\mathbb{R}^d$, totaling about 60,000 in this case, a sparsity parameter $k$, and an upper bound $m$, find a set of unit vectors $A_1, A_2, \ldots, A_m$ such that

$$v_w = \sum_{j=1}^{m} \alpha_{w,j} A_j + \eta_w$$  \hspace{1cm} (3)

where at most $k$ of the coefficients $\alpha_{w,1}, \ldots, \alpha_{w,m}$ are nonzero (so-called hard sparsity constraint), and $\eta_w$ is a noise vector.
Extracting Word Senses: K-SVD

\[
\sum_{w} |v_w - \sum_{j=1}^{m} \alpha_{w,j} A_j|^2.
\]

Or

\[
\min_{D,X} \{ \|Y - DX\|_F^2 \} \quad \text{subject to} \quad \forall i, \|x_i\|_0 = 1.
\]
Results
- 乒乓球, 羽毛球, 跳水, 游泳, 举重
- Ping pong, Badminton, Diving, Swimming, Lifting
- 藏羚羊, 大熊猫, 娃娃鱼, 金丝猴, 小熊猫
- Tibetan Antelope, Panda, Chinese Giant Salamander, golden monkey, small panda
- 人, 球, 大, 小, 一
- Person, Ball, Big, Small, One
Use K-SVD to convert base embedding into atoms of discourses, and train with discourse vector as label.
Challenge in Experiment

- Takes up a huge amount of memory
- Segmentation Fault on big embeddings
- Non-Convex problem
- A lot of manual work required for preparing training data that match definition with word sense in terms of discourse vector.
Discussion
Dynamic Embedding Tool

- Word Embedding Tools:
  - Add
  - Swap
  - Filter/Delete
Possible Application: Chat

- Add: ask for clarification on novel concept/words and encode reply into word embedding to maintain conversational context.
- Swap: Convert polysemy words into multiple vectors to improve semantic accuracy
- Filter: Control grammar, rhyme, mood.
Related Work: Sememe (ACL 2017)

- Sememe: unit meaning
- HowNet (Chinese)
- Compose unit meaning to improve word embedding
- Model: attention over context words or target
Related Work: Jump LSTM (ACL 2017)

- Mimic skimming
- Trained using reinforcement learning
- Improved accuracy of LSTM over long text.

<table>
<thead>
<tr>
<th>Seq length</th>
<th>LSTM-Jump</th>
<th>LSTM</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test accuracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>98%</td>
<td>96%</td>
<td>n/a</td>
</tr>
<tr>
<td>100</td>
<td>98%</td>
<td>96%</td>
<td>n/a</td>
</tr>
<tr>
<td>1000</td>
<td>90%</td>
<td>80%</td>
<td>n/a</td>
</tr>
<tr>
<td>Test time (Avg tokens read)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13.5s (2.1)</td>
<td>18.9s (10)</td>
<td>1.40x</td>
</tr>
<tr>
<td>100</td>
<td>13.9s (2.2)</td>
<td>120.4s (100)</td>
<td>8.66x</td>
</tr>
<tr>
<td>1000</td>
<td>18.9s (3.0)</td>
<td>1250s (1000)</td>
<td>66.14x</td>
</tr>
</tbody>
</table>
Related Work: Elastic Weight Consolidation

- Prevent forgetting, continual learning using experience from related tasks
- Intuitively similar to human brain’s synaptic consolidation (proven neuroscience concept).
Special thanks to:

- Professor Li
- Junnan Chen
- Wordnik.com
- Presentation template by SlidesCarnival
THANKS!

Any questions?