StarSpace: Embed All The Things

Jan Pichl
Motivation

- Dimensionality reduction
- Incorporating additional knowledge
- One embedding for multiple problems:
  - Text classification, ranking, image labelling, embedding of graphs (words, sentences, documents)
- One embedding for various types of entities
  - Comparing different types of entities - sentences, graphs, images, words, ...
Unsupervised method for words - word2vec

- We need to embed a word into a lower dimensional space
- Skip-gram neural network
- Arithmetic operations show some interesting relations
Word2vec in matrix form

- The trained matrix is used as a lookup table
Generalization

- We need to embed other entities than words
- We need to embed different kinds of entities into a same vector space
- Compare embeddings vectors of different kind of entities
- Train a lookup table similar to a word2vec
- Using a supervised task to train the representation
StarSpace

- Facebook research
- Star (*) Space - embedding for “everything” -> Star
- Main idea
  - Transform discrete features to a vector of real numbers
  - Use one simple approach for different task (avoid computationally expensive neural networks)
- Directly trained on a desired task or...
- The model can be used as an input for neural network for different tasks
● Input: discrete feature vector
● Each entity consists of one or more features (bag of features)
● Embedding matrix: $D \times d$ (number of features x desired dimensionality)
● Each row of a matrix is the embedding for one feature
● Entity with multiple features - sum of embeddings
● Fixed feature dictionary
Optimization

- Similarity function: cosine similarity
- Loss function:

\[
\sum_{(a,b) \in E^+} L^{batch}(\text{sim}(a, b), \text{sim}(a, b^-), \ldots, \text{sim}(a, b^-_k))
\]

E+: positive entity pairs
E-: negative entity
(negative sampling)

- Positive and negative sample
- Task specific sample generation

Embed of entity
- dot product
- cosine sim

ranking loss
Training

- Stochastic Gradient Descent
- Adagrad
- Each SGD step contains one positive sample
- Margin ranking loss:
  - \( \max \{0, \mu - \text{sim}(a, b) + \text{sim}(a, b^-)\} \)
- Positive examples need to be ranked higher than negative ones
- Each batch contains \( k \) negative examples

Testing

- Direct usage of similarity function
- Embedding as an input for another model
Tasks

- **Classification:**
  - Positive: training set
  - Negative: labels sampled from set of labels

- **Collaborative Filtering-based Recommendation:**
  - Positive: user ID embedding OR list of items users like except one, label - the left out item
  - Negative: sampled from possible items

- **Content-based Recommendation:**
  - User: bag of documents, Document: bag of words
  - Same as before: one left out

- **Multi-Relational Knowledge Graphs:**
  - Graph (head, relation, tail) e.g. (Beyonce, born-in, Houston)
  - Randomly remove head or tail -> removed item as a label
E+: [‘restaurant has great food’, ‘#yum #restaurant’]
E-: [‘#animals’, ‘#donald trump’]

The higher the ranking of the relevant item the better

In Testing:
for a given input \(a\), using \(\max_{\hat{b}} \text{sim}(a, \hat{b})\)
Tasks

- **Document Search:**
  - Input: query keywords, Output: document bag of words
  - Unsupervised data: random keywords from document
  - Negative: irrelevant documents
  - Implicit learning of document embeddings

- **Word embeddings:**
  - Input: windows of words (n words on both sides of selected word)
  - Output: selected word

- **Sentence embeddings:**
  - Positive pairs: sentences from the same document
  - Negative: from different documents

- **Image labelling:**
  - Input: ResNet (or another model) image features
  - Output: Image label
  - Negative: irrelevant labels
Number of k negative samples

- Dataset - collection of Freebase triplets
- Removed head and tail respectively
- Prediction of missing entity
- Reported how many entities were ranked among first ten (hit@10)
- Best result on this task: k = 50

<table>
<thead>
<tr>
<th>K</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>250</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Epochs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3260</td>
<td>711</td>
<td>318</td>
<td>130</td>
<td>69</td>
<td>34</td>
<td>13</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>67.05%</td>
<td>68.08%</td>
<td>68.13%</td>
<td>67.63%</td>
<td>69.05%</td>
<td>66.99%</td>
<td>63.95%</td>
<td>60.32%</td>
<td>54.14%</td>
</tr>
</tbody>
</table>

Table 5: Adapting the number of negative samples k for a 50-dim model for 1 hour of training on Freebase 15k.
Results

- Tasks: select sentence from Wikipedia article, try to find the article it came from
- Original article has sentences as a features (minus the original one)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Hits@1</th>
<th>Hits@10</th>
<th>Hits@20</th>
<th>Mean Rank</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFIDF</td>
<td>56.63%</td>
<td>72.80%</td>
<td>76.16%</td>
<td>578.98</td>
<td>-</td>
</tr>
<tr>
<td>fastText (public Wikipedia model)</td>
<td>18.08%</td>
<td>36.36%</td>
<td>42.97%</td>
<td>987.27</td>
<td>-</td>
</tr>
<tr>
<td>fastText (our dataset)</td>
<td>16.89%</td>
<td>37.60%</td>
<td>45.25%</td>
<td>786.77</td>
<td>40h</td>
</tr>
<tr>
<td>Supervised method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM Ranker BoW features</td>
<td>56.73%</td>
<td>69.24%</td>
<td>71.86%</td>
<td>723.47</td>
<td>-</td>
</tr>
<tr>
<td>SVM Ranker: fastText features (public)</td>
<td>18.44%</td>
<td>37.80%</td>
<td>45.91%</td>
<td>887.96</td>
<td>-</td>
</tr>
<tr>
<td>StarSpace</td>
<td>56.75%</td>
<td>78.14%</td>
<td>83.15%</td>
<td>122.26</td>
<td>89h</td>
</tr>
</tbody>
</table>
### Results

- **Tasks:** select two sentences from a single Wikipedia article
- **Based on the one sentence we want to try the other one

<table>
<thead>
<tr>
<th>Metric</th>
<th>Hits@1</th>
<th>Hits@10</th>
<th>Hits@20</th>
<th>Mean Rank</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsupervised methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFIDF</td>
<td>24.79%</td>
<td>35.53%</td>
<td>38.25%</td>
<td>2523.68</td>
<td>-</td>
</tr>
<tr>
<td>fastText (public Wikipedia model)</td>
<td>5.77%</td>
<td>14.08%</td>
<td>17.79%</td>
<td>2393.38</td>
<td>-</td>
</tr>
<tr>
<td>fastText (our dataset)</td>
<td>5.47%</td>
<td>13.54%</td>
<td>17.60%</td>
<td>2363.74</td>
<td>40h</td>
</tr>
<tr>
<td>StarSpace (word-level training)</td>
<td>5.89%</td>
<td>16.41%</td>
<td>20.60%</td>
<td>1614.21</td>
<td>45h</td>
</tr>
<tr>
<td><strong>Supervised methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM Ranker BoW features</td>
<td>26.36%</td>
<td>36.48%</td>
<td>39.25%</td>
<td>2368.37</td>
<td>-</td>
</tr>
<tr>
<td>SVM Ranker: fastText features (public)</td>
<td>5.81%</td>
<td>12.14%</td>
<td>15.20%</td>
<td>1442.05</td>
<td>-</td>
</tr>
<tr>
<td>StarSpace (sentence pair training)</td>
<td>30.07%</td>
<td>50.89%</td>
<td>57.60%</td>
<td>422.00</td>
<td>36h</td>
</tr>
<tr>
<td>StarSpace (word+sentence training)</td>
<td>25.54%</td>
<td>45.21%</td>
<td>52.08%</td>
<td>484.27</td>
<td>69h</td>
</tr>
</tbody>
</table>
Thank you!

https://www.slideshare.net/akihikowatanabe3110/starspace-embed-all-the-things