Embedding Methods for NLP
Part 1: Unsupervised and Supervised Embeddings

Jason Weston & Antoine Bordes
Facebook AI Research

EMNLP tutorial – October 29, 2014
What is a word embedding?

Suppose you have a dictionary of words.

The $i^{th}$ word in the dictionary is represented by an embedding:

$$w_i \in \mathbb{R}^d$$

i.e. a $d$-dimensional vector, which is learnt!

- $d$ typically in the range 50 to 1000.
- Similar words should have similar embeddings (share latent features).
- Embeddings can also be applied to symbols as well as words (e.g. Freebase nodes and edges).
- Discuss later: can also have embeddings of phrases, sentences, documents, or even other modalities such as images.
Learning an Embedding Space

Example of Embedding of 115 Countries (Bordes et al., ’11)
Main methods we highlight, ordered by date.

- Latent Semantic Indexing (Deerwester et al., ’88).
- Neural Net Language Models (NN-LMs) (Bengio et al., ’06).
- Convolutional Nets for tagging (SENNNA) (Collobert & Weston, ’08).
- Supervised Semantic Indexing (Bai et al, ’09).
- Wsabie (Weston et al., ’10).
- Recurrent NN-LMs (Mikolov et al., ’10).
- Recursive NNs (Socher et al., ’11).
- Word2Vec (Mikolov et al., ’13).
- Paragraph Vector (Le & Mikolov, ’14).
- Overview of recent applications.
Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:
- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, ’09).
- Wsabie (Weston et al., ’10).

Embeddings for Language Modeling (useful for speech, translation, ...):
- Neural Net Language Models (NN-LMs) (Bengio et al., ’06)
- Recurrent NN-LMs (Mikolov et al., ’10).
- Word2Vec (Mikolov et al., ’13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):
- Convolutional Nets for tagging (SENNNA) (Collobert & Weston, ’08).
- Recursive NNs (Socher et al., ’11).
- Paragraph Vector (Le & Mikolov, ’14).
Main methods we highlight, ordered by topic.

**Embeddings for Ranking and Retrieval:**
- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al, '09).
- Wsabie (Weston et al., '10).

**Embeddings for Language Modeling (useful for speech, translation, . . .):**
- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

**Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):**
- Convolutional Nets for tagging (SENNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).
We want to learn to match a query (text) to a target (text).

Many classical supervised ranking methods use hand-coded features.

Methods like LSI that learn from words are unsupervised.

Supervised Semantic Indexing (SSI) uses supervised learning from text only:


Outperforms existing methods (on words) like TFIDF, LSI or a (supervised) margin ranking perceptron baseline.
Basic Bag-O’-Words

Bag-of-words + cosine similarity:

- Each doc. \( \{d_t\}_{t=1}^{N} \subset \mathbb{R}^{D} \) is a *normalized* bag-of-words.
- Similarity with query \( q \) is: \( f(q, d) = q^\top d \)

Doesn’t deal with synonyms: bag vectors can be orthogonal

*No machine learning at all*
Latent semantic indexing (LSI)

Learn a linear embedding $\phi(d_i) = Ud_i$ via a reconstruction objective. Rank with: $f(q, d) = q^T U^T Ud = \phi(q)^T \phi(d_i)$ \(^1\).

Uses “synonyms”: low-dimensional latent “concepts”.

Unsupervised machine learning: useful for goal?

\[^1f(q, d) = q^T (U^T U + \alpha I)d\] gives better results. Also, usually normalize this $\to$ cosine similarity.
Supervised Semantic Indexing (SSI)

- Basic model: rank with
  \[ f(q, d) = q^\top Wd = \sum_{i,j=1}^{D} q_i W_{ij} d_j \]
  i.e. learn weights of polynomial terms between documents.

- Learn \( W \in \mathbb{R}^{D \times D} \) (huge!) with click-through data or other labels.

  Uses “synonyms”

  Supervised machine learning: targeted for goal

  Too Big/Slow?! Solution = Constrain \( W \):
  low rank \( \rightarrow \) embedding model!
SSI: why is this a good model?

Classical bag-of-words doesn't work when there are few matching terms:

\( q = (\text{kitten, vet, nyc}) \)

\( d = (\text{cat, veterinarian, new, york}) \)

Method \( q^T Wd \) learns that e.g. \( \text{kitten} \) and \( \text{cat} \) are highly related.

E.g. if \( i \) is the index of \( \text{kitten} \) and \( j \) is the index of \( \text{cat} \), then \( W_{ij} > 0 \) after training.
**SSI: Why the Basic Model Sucks**

*W* is big: 3.4Gb if $D = 30000$, 14.5Tb if $D = 2.5M$.

Slow: $\mathbf{q}^\top \mathbf{W} \mathbf{d}$ computation has $mn$ computations $q_j W_{ij} d_i$, where *q* and *d* have *m* and *n* nonzero terms.

Or one computes $\mathbf{v} = \mathbf{q}^\top \mathbf{W}$ once, and then $\mathbf{v} \mathbf{d}$ for each document. Classical speed where query has $D$ terms, assuming $\mathbf{W}$ is dense $\rightarrow$ still slow.

One could minimize $\|\mathbf{W}\|_1$ and attempt to make $\mathbf{W}$ sparse. Then at most $mp$ times slower than classical model (with *p* nonzeros in a column.)
SSI Improved model: Low Rank $W$

**Constrain $W$:**

$$W = U^T V + I.$$ 

- $U$ and $V$ are $N \times D$ matrices $\rightarrow$ smaller
- Low dimensional “latent concept” space like LSI (same speed).
- Differences: supervised, asymmetric, learns with $I$.

Variants:

- $W = I$: bag-of-words again.
- $W = D$, reweighted bag-of-words related to [Grangier and Bengio, 2005].
- $W = U^T U + I$: symmetric.
SSI: Training via maximizing AUC

- Given a set of tuples \( R \) with a query \( q \), a related document \( d^+ \) and an unrelated (or lower ranked) document \( d^- \).
- We would like \( f(q, d^+) > f(q, d^-) \).
- Minimize margin ranking loss [Herbrich et al., 2000]:

\[
\sum_{(q,d^+,d^-) \in R} \max(0, 1 - f(q, d^+) + f(q, d^-)).
\]

Learning Algorithm Stochastic Gradient Descent: Fast & scalable.

<table>
<thead>
<tr>
<th>Iterate</th>
<th>Sample a triplet ((q, d^+, d^-)),</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update ( W \leftarrow W - \lambda \frac{\partial}{\partial W} \max(0, 1 - f(q, d^+) + f(q, d^-)) ).</td>
</tr>
</tbody>
</table>

Other options: batch gradient, parallel SGD (hogwild), Adagrad . . .
Training: setting hyperparameters

The following hyperparameters can be tuned for training:

- The initial random weights of the embedding vectors:
  e.g. use (mean 0, variance $\frac{1}{\sqrt{d}}$).

- The learning rate (typically: 0.0001, 0.001, 0.01, 0.1, ...).

- The value of the margin (e.g.: 1, 0.5, 0.2, 0.1, ...).

- Restricting the norm of embeddings:
  $\|U_i\| \leq C$, $\|V_i\| \leq C$ (e.g.: $C=1$).

All these parameters are relative to each other, e.g. a larger margin might need larger initial weights and learning rate.

Typically, we fix the initialization and norm, and try different values of margin and learning rate. This can make big differences in performance.
Prior Work: Summary of learning to Rank

- [Grangier & Bengio, '06] used similar methods to basic SSI for retrieving images.
- [Goel, Langord & Strehl, '08] used Hash Kernels (Vowpal Wabbit) for advert placement.
- Main difference: SSI uses low rank on $W$.
- SVM [Joachims, 2002] and NN ranking methods [Burges, 2005].
  Use hand-coded features: title, body, URL, search rankings, ... (don’t use words)
  (e.g. Burges uses 569 features in all).
- In contrast SSI uses only the words and trains on huge feature sets.
- Several works on optimizing different loss functions (MAP, ROC, NDCG): [Cao, 2008], [Yu, 2007], [Qin, 2006], ...
- Lots of stuff for “metric learning” problem as well.

One could also add features + new loss to this method ..
Experimental Comparison

- **Wikipedia**
  - 1,828,645 documents. 24,667,286 links.
  - Split into 70% train, 30% test.

- Pick random doc. as query, then rank other docs.
- Docs that are linked to it should be highly ranked.

**Two setups:**
- (i) whole document is used as query;
- (ii) 5, 10 or 20 words are picked to mimic keyword search.
Wikipedia Experiments: Document Retrieval Performance

Experiments on Wikipedia, which contains 1.8M documents: retrieval task using the link structure and separated the data into 70% for training and 30% for test.

### Document based retrieval:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rank-Loss</th>
<th>MAP</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFIDF</td>
<td>0.842%</td>
<td>0.432±0.012</td>
<td>0.193</td>
</tr>
<tr>
<td>$\alpha$LSI + $(1 - \alpha)$TFIDF</td>
<td>0.721%</td>
<td>0.433</td>
<td>0.193</td>
</tr>
<tr>
<td>Linear SVM Ranker</td>
<td>0.410%</td>
<td>0.477</td>
<td>0.212</td>
</tr>
<tr>
<td>Hash Kernels + $\alpha I$</td>
<td>0.322%</td>
<td>0.492</td>
<td>0.215</td>
</tr>
<tr>
<td>SSI</td>
<td>0.158%</td>
<td>0.547±0.012</td>
<td>0.239±0.008</td>
</tr>
</tbody>
</table>

### $k$-keywords based retrieval:

<table>
<thead>
<tr>
<th>$k = 5$: Algorithm</th>
<th>Params</th>
<th>Rank</th>
<th>MAP</th>
<th>P@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFIDF</td>
<td>0</td>
<td>21.6%</td>
<td>0.047</td>
<td>0.023</td>
</tr>
<tr>
<td>$\alpha$LSI + $(1 - \alpha)$TFIDF</td>
<td>200$D$ + 1</td>
<td>14.2%</td>
<td>0.049</td>
<td>0.023</td>
</tr>
<tr>
<td>SSI</td>
<td>400$D$</td>
<td>4.37%</td>
<td>0.166</td>
<td>0.083</td>
</tr>
</tbody>
</table>
Scatter Plots: SSI vs. TFIDF and LSI

Figure: Scatter plots of Average Precision for 500 documents: (a) SSI vs. TFIDF, (b) SSI vs. $\alpha \text{LSI} + (1-\alpha) \text{TFIDF}$. 
Experiments: Cross-Language Retrieval

Retrieval experiments using a query document in Japanese, where the task is to retrieve documents in English (using link structure as ground truth).

SSI can do this without doing a translation step first as it learns to map the two languages together in the embedding space.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rank-Loss</th>
<th>MAP</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFIDF&lt;sub&gt;EngEng&lt;/sub&gt; (Google translated queries)</td>
<td>4.78%</td>
<td>0.319</td>
<td>0.259</td>
</tr>
<tr>
<td>(\alpha)LSI&lt;sub&gt;EngEng&lt;/sub&gt; (+(1 - \alpha))TFIDF&lt;sub&gt;EngEng&lt;/sub&gt;</td>
<td>3.71%</td>
<td>0.300</td>
<td>0.253</td>
</tr>
<tr>
<td>(\alpha)CL-LSI&lt;sub&gt;JapEng&lt;/sub&gt; (+(1 - \alpha))TFIDF&lt;sub&gt;EngEng&lt;/sub&gt;</td>
<td>3.31%</td>
<td>0.275</td>
<td>0.212</td>
</tr>
<tr>
<td>SSI&lt;sub&gt;EngEng&lt;/sub&gt;</td>
<td>1.72%</td>
<td>0.399</td>
<td>0.325</td>
</tr>
<tr>
<td>SSI&lt;sub&gt;JapEng&lt;/sub&gt;</td>
<td>0.96%</td>
<td>0.438</td>
<td>0.351</td>
</tr>
<tr>
<td>(\alpha)SSI&lt;sub&gt;JapEng&lt;/sub&gt; (+(1 - \alpha))TFIDF&lt;sub&gt;EngEng&lt;/sub&gt;</td>
<td>0.75%</td>
<td>0.493</td>
<td>0.377</td>
</tr>
<tr>
<td>(\alpha)SSI&lt;sub&gt;JapEng&lt;/sub&gt; (+(1 - \alpha))SSI&lt;sub&gt;EngEng&lt;/sub&gt;</td>
<td>0.63%</td>
<td>0.524</td>
<td>0.386</td>
</tr>
</tbody>
</table>

Some recent related translation-based embeddings: (Hermann & Blunsom, ICLR '14) and (Mikolov et al., '13).
Wsabie (Weston, Bengio & Usunier, ’10)

- Extension to SSI, also embeds objects other than text, e.g. images.
- WARP loss function that optimizes precision@k.

Learn $\Phi_1(\cdot)$ and $\Phi_w(\cdot)$ to optimize precision@k.
Joint Item-Item Embedding Model

L.H.S: Image, query string or user profile (depending on the task)

\[ \Phi_{LHS}(x) = U\Phi_x(x) : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{100}. \]

R.H.S: document, image, video or annotation (depending on the task)

\[ \Phi_{RHS}(y) = V\Phi_y(y) : \mathbb{R}^{d_y} \rightarrow \mathbb{R}^{100}. \]

This model again compares the degree of match between the L.H.S and R.H.S in the embedding space:

\[ f_y(x) = \text{sim}(\Phi_{LHS}(x), \Phi_{RHS}(y)) = \Phi_x(x)^\top U^\top V\Phi_y(y) \]

Also constrain the weights (regularize):

\[ \|U_i\|_2 \leq C, \quad i = 1, \ldots, d_x, \quad \|V_i\|_2 \leq C, \quad i = 1, \ldots, d_y. \]
Classical approach to learning to rank is maximize AUC by minimizing:

\[
\sum_{x} \sum_{y} \sum_{\bar{y} \neq y} \max(0, 1 + f_{\bar{y}}(x) - f_{y}(x))
\]

**Problem:** All pairwise errors are considered the same, it counts the number of ranking violations.

**Example:**
- Function 1: true annotations ranked 1st and 101st.
- Function 2: true annotations ranked 50th and 52nd.
  
AUC prefers these *equally* as both have 100 “violations”.

We want to optimize the top of the ranked list!
Rank Weighted Loss [Usunier et al. '09]

Replace classical AUC optimization:

\[
\sum_x \sum_y \sum_{\bar{y} \neq y} \max(0, 1 + f_{\bar{y}}(x) - f_y(x))
\]

With weighted version:

\[
\sum_x \sum_y \sum_{\bar{y} \neq y} L(rank_{\bar{y}}(x)) \max(0, 1 + f_{\bar{y}}(x) - f_y(x))
\]

where \( rank_{\bar{y}}(f(x)) \) is the rank of the true label:

\[
rank_{\bar{y}}(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) \geq f_y(x))
\]

and \( L(\eta) \) converts the rank to a weight, e.g. \( L(\eta) = \sum_{i=1}^{\eta} 1/\eta \).
Weighted Approximate-Rank Pairwise (WARP) Loss

**Problem:** we would like to apply SGD:

$$\text{Weighting } L(\text{rank}_y(f(x))), \quad \text{rank}_y(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) + 1 \geq f_y(x))$$

... too expensive to compute per \((x, y)\) sample as \(y \in \mathcal{Y}\) is large.

**Solution:** approximate by sampling \(f_i(x)\) until we find a violating label \(\bar{y}\)

$$\text{rank}_y(f(x)) \approx \left\lfloor \frac{|\mathcal{Y}| - 1}{N} \right\rfloor$$

where \(N\) is the number of trials in the sampling step.
Online WARP Loss

**Input:** labeled data \((x_i, y_i), \ y_i \in \{1, \ldots, Y\}\).

repeat

Pick a random labeled example \((x_i, y_i)\)

Set \(N = 0\).

repeat

Pick a random annotation \(\bar{y} \in \{1, \ldots, Y\} \setminus y_i\).

\(N = N + 1\).

until \(f_{\bar{y}}(x) > f_{y_i}(x) - 1\) or \(N > Y - 1\)

if \(f_{\bar{y}}(x) > f_{y_i}(x) - 1\) then

Make a gradient step to minimize:

\[ L(\left\lfloor \frac{Y-1}{N} \right\rfloor)|1 - f_{\bar{y}}(x) + f_{y_i}(x)|_+ \]

end if

until validation error does not improve.
### Image Annotation Performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>16k ImageNet</th>
<th>22k ImageNet</th>
<th>97k Web Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Means</td>
<td>4.4%</td>
<td>2.7%</td>
<td>2.3%</td>
</tr>
<tr>
<td>One-vs-all SVMs 1+:1-</td>
<td>4.1%</td>
<td>3.5%</td>
<td>1.6%</td>
</tr>
<tr>
<td>One-vs-all SVMs</td>
<td>9.4%</td>
<td>8.2%</td>
<td>6.8%</td>
</tr>
<tr>
<td>AUC SVM Ranker</td>
<td>4.7%</td>
<td>5.1%</td>
<td>3.1%</td>
</tr>
<tr>
<td>Wsabie</td>
<td>11.9%</td>
<td>10.5%</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

### Training time: WARP vs. OWPC-SGD & AUC

![Graph showing training time comparison between WARP, AUC, and OWPC-SGD on ImageNet](image-url)
Learned Annotation Embedding (on Web Data)

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Neighboring Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>barack obama, david beckham, santa</td>
<td>barak obama, obama, barack, barrack obama, bow wow beckham, david beckam, alessandro del piero, del piero santa claus, papa noel, pere noel, santa clause, joyeux noel</td>
</tr>
<tr>
<td>dolphin, cows</td>
<td>delphin, dauphin, whale, delfin, delfini, baleine, blue whale cattle, shire, dairy cows, kuh, horse, cow, shire horse, kone</td>
</tr>
<tr>
<td>rose, pine tree</td>
<td>rosen, hibiscus, rose flower, rosa, roze, pink rose, red rose abies alba, abies, araucaria, pine, neem tree, oak tree</td>
</tr>
<tr>
<td>mount fuji, eiffel tower</td>
<td>mt fuji, fuji, fujisan, fujiyama, mountain, zugspitze eiffel, tour eiffel, la tour eiffel, big ben, paris, blue mosque</td>
</tr>
<tr>
<td>ipod, f18</td>
<td>ipod, ipod nano, apple ipod, ipod apple, new ipod f 18, eurofighter, f14, fighter jet, tomcat, mig 21, f 16</td>
</tr>
</tbody>
</table>
Summary

Conclusion

- **Powerful**: supervised methods for ranking.
- **Outperform** classical methods
- **Efficient** low-rank models → learn hidden representations.

Embeddings good for generalization, but can “blur” too much e.g. for exact word matches.

Extensions

- Nonlinear extensions – e.g. convolutional net instead.
  e.g. DeViSE (Frome et al., NIPS ’13)
Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:
- Latent Semantic Indexing (Deerwester et al., ’88).
- Supervised Semantic Indexing (Bai et al., ’09).
- Wsabie (Weston et al., ’10).

Embeddings for Language Modeling (useful for speech, translation)
- Neural Net Language Models (NN-LMs) (Bengio et al., ’06)
- Recurrent NN-LMs (Mikolov et al., ’10).
- Word2Vec (Mikolov et al., ’13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):
- Convolutional Nets for tagging (SENNA) (Collobert & Weston, ’08).
- Recursive NNs (Socher et al., ’11).
- Paragraph Vector (Le & Mikolov, ’14).
Language Modeling

Task: given a sequence of words, predict the next word.

the cat sat on the ??

- $n$-gram models are a strong baseline on this task.
- A variety of embedding models have been tried, they can improve results.
- The embeddings learnt from this unsupervised task can also be used to transfer to and improve a supervised task.
Neural Network Language Models

Neural Network Language Models: Hierarchical Soft Max Trick (Morin & Bengio ’05)

Predicting the probability of each next word is slow in NNLMs because the output layer of the network is the size of the dictionary.

Can predict via a tree instead:

1. Cluster the dictionary either according to semantics (similar words in the same cluster) or frequency (common words in the same cluster). *This gives a two-layer tree, but a binary tree is another possibility.*
2. The internal nodes explicitly model the probability of its child nodes.
3. The cost of predicting the probability of the true word is now: traversal to the child, plus normalization via the internal nodes and children in the same node.

This idea is used in Word2Vec and RNN models as well.
Recurrent Neural Network Language Models

**Key idea:** *input to predict next word is current word plus context fed-back from previous word (i.e. remembers the past with recurrent connection).*

![Recurrent neural network based LM](image)

**Figure:** *Recurrent neural network based LM*

Recurrent neural network based language model. Mikolov et al., Interspeech, ’10.
Recent uses of NNLMs and RNNs to improve machine translation:
Also (Kalchbrenner ’13), (Sutskever et al., ’14), (Cho et al., ’14).
Word2Vec: very simple LM, works well


CBOw

Skip-gram
Word2Vec: compositionality

Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-gram [32]</td>
<td>39</td>
</tr>
<tr>
<td>Average LSA similarity [32]</td>
<td>49</td>
</tr>
<tr>
<td>Log-bilinear model [24]</td>
<td>54.8</td>
</tr>
<tr>
<td>RNNLMs [19]</td>
<td>55.4</td>
</tr>
<tr>
<td>Skip-gram</td>
<td>48.0</td>
</tr>
<tr>
<td>Skip-gram + RNNLMs</td>
<td><strong>58.9</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Czech + currency</th>
<th>Vietnam + capital</th>
<th>German + airlines</th>
<th>Russian + river</th>
<th>French + actress</th>
</tr>
</thead>
<tbody>
<tr>
<td>koruna</td>
<td>Hanoi</td>
<td>airline Lufthansa</td>
<td>Moscow</td>
<td>Juliette Binoche</td>
</tr>
<tr>
<td>Check crown</td>
<td>Ho Chi Minh City</td>
<td>carrier Lufthansa</td>
<td>Volga River</td>
<td>Vanessa Paradis</td>
</tr>
<tr>
<td>Polish zolty</td>
<td>Viet Nam</td>
<td>flag carrier Lufthansa</td>
<td>upriver</td>
<td>Charlotte Gainsbourg</td>
</tr>
<tr>
<td>CTK</td>
<td>Vietnamese</td>
<td>Lufthansa</td>
<td>Russia</td>
<td>Cecile De</td>
</tr>
</tbody>
</table>

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Code: [https://code.google.com/p/word2vec/](https://code.google.com/p/word2vec/)
Main methods we highlight, ordered by topic.

Embeddings for Ranking and Retrieval:
- Latent Semantic Indexing (Deerwester et al., '88).
- Supervised Semantic Indexing (Bai et al., '09).
- Wsabie (Weston et al., '10).

Embeddings for Language Modeling (useful for speech, translation, ...):
- Neural Net Language Models (NN-LMs) (Bengio et al., '06)
- Recurrent NN-LMs (Mikolov et al., '10).
- Word2Vec (Mikolov et al., '13).

Embeddings for Supervised Prediction Tasks (POS, chunk, NER, SRL, sentiment, etc.):
- Convolutional Nets for tagging (SENNNA) (Collobert & Weston, '08).
- Recursive NNs (Socher et al., '11).
- Paragraph Vector (Le & Mikolov, '14).
NLP Tasks

- Part-Of-Speech Tagging (POS): syntactic roles (noun, adverb...)
- Chunking: syntactic constituents (noun phrase, verb phrase...)
- Name Entity Recognition (NER): person/company/location...
- Semantic Role Labeling (SRL):
  
  
  [John] \textit{ARG0} [ate] \textit{REL} [the apple] \textit{ARG1} [in the garden] \textit{ARGM-LOC}

Complex Systems

- Two extreme choices to get a complex system

  - Large Scale Engineering: design a lot of complex features, use a fast existing linear machine learning algorithm

  - Large Scale Machine Learning: use simple features, design a complex model which will implicitly learn the right features
The Large Scale Feature Engineering Way

- Extract **hand-made features** e.g. from the parse tree
- **Disjoint**: all tasks trained separately, **Cascade** features
- Feed these features to a **shallow** classifier like **SVM**
**ASSERT:** many hand built features for SRL (Pradhan et al, ’04)

Problems:
1) Features rely on other solutions *(parsing, named entity, word-sense)*
2) Technology task-transfer is *difficult*

- Choose some **good hand-crafted features**

<table>
<thead>
<tr>
<th>Features</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate and POS tag of predicate</td>
<td>Voice: active or passive (hand-built rules)</td>
</tr>
<tr>
<td>Phrase type: adverbial phrase, prepositional phrase, ...</td>
<td>Governing category: Parent node’s phrase type(s)</td>
</tr>
<tr>
<td>Head word and POS tag of the head word</td>
<td>Position: left or right of verb</td>
</tr>
<tr>
<td>Path: traversal from predicate to constituent</td>
<td>Predicted named entity class</td>
</tr>
<tr>
<td>Word-sense disambiguation of the verb</td>
<td>Verb clustering</td>
</tr>
<tr>
<td>Length of the target constituent (number of words)</td>
<td>NEG feature: whether the verb chunk has a ”not”</td>
</tr>
<tr>
<td>Partial Path: lowest common ancestor in path</td>
<td>Head word replacement in prepositional phrases</td>
</tr>
<tr>
<td>First and last words and POS in constituents</td>
<td>Ordinal position from predicate + constituent type</td>
</tr>
<tr>
<td>Constituent tree distance</td>
<td>Temporal cue words (hand-built rules)</td>
</tr>
<tr>
<td>Dynamic class context: previous node labels</td>
<td>Constituent relative features: phrase type</td>
</tr>
<tr>
<td>Constituent relative features: head word</td>
<td>Constituent relative features: head word POS</td>
</tr>
<tr>
<td>Constituent relative features: siblings</td>
<td>Number of pirates existing in the world...</td>
</tr>
</tbody>
</table>

- Feed them to a **shallow classifier** like SVM
The Suboptimal (?) Cascade

(Or, the opposing view is the above is a smart use of prior knowledge..)
Goals

- Task-specific engineering limits NLP scope
- Can we find unified hidden representations?
- Can we build unified NLP architecture?

Means

- Start from scratch: forget (most of) NLP knowledge
- Compare against classical NLP benchmarks
- Our dogma: avoid task-specific engineering
## NLP Benchmarks

- **Datasets:**
  - POS, CHUNK, SRL: **WSJ** (≈ up to 1M labeled words)
  - NER: **Reuters** (≈ 200K labeled words)

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shen, 2007</td>
<td>97.33%</td>
</tr>
<tr>
<td><strong>Toutanova, 2003</strong></td>
<td>97.24%</td>
</tr>
<tr>
<td>Gimenez, 2004</td>
<td>97.16%</td>
</tr>
</tbody>
</table>

  (a) **POS**: As in (Toutanova, 2003)

<table>
<thead>
<tr>
<th>System</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ando, 2005</td>
<td>89.31%</td>
</tr>
<tr>
<td>Florian, 2003</td>
<td>88.76%</td>
</tr>
<tr>
<td>Kudoh, 2001</td>
<td>88.31%</td>
</tr>
</tbody>
</table>

  (c) **NER**: CoNLL 2003

<table>
<thead>
<tr>
<th>System</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sha, 2003</strong></td>
<td>94.29%</td>
</tr>
<tr>
<td>Kudoh, 2001</td>
<td>93.91%</td>
</tr>
</tbody>
</table>

  (b) **CHUNK**: CoNLL 2000

<table>
<thead>
<tr>
<th>System</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koomen, 2005</td>
<td>77.92%</td>
</tr>
<tr>
<td>Pradhan, 2005</td>
<td>77.30%</td>
</tr>
<tr>
<td>Haghighi, 2005</td>
<td>77.04%</td>
</tr>
</tbody>
</table>

  (d) **SRL**: CoNLL 2005

- We chose as **benchmark systems**:
  - Well-established systems
  - Systems avoiding external labeled data

- **Notes**:
  - Ando, 2005 uses external unlabeled data
  - Koomen, 2005 uses 4 parse trees not provided by the challenge
The “Deep Learning” Way

Neural nets attempt to propose a radically different end-to-end approach:

- Avoid building a parse tree. Humans don’t need this to talk.
- Try to avoid all hand-built features → monolithic systems.
Neural Networks

• **Stack** several layers together

\[
\begin{align*}
\text{Input Vector} & \quad x \\
\text{Matrix-vector operation} & \quad W_1x \\
\text{Non-Linearity} & \quad \text{HardTanh} \\
\text{Matrix-vector operation} & \quad W_2 \\
\text{Output Vector} & \quad y
\end{align*}
\]

• **Increasing** level of abstraction at each layer

• Requires **simpler features** than “shallow” classifiers

• The “weights” \( W_i \) are trained by gradient descent

• How can we feed words?
The Big Picture

A unified architecture for all NLP (labeling) tasks:

**Sentence:**

- Felix
- sat
- on
- the
- mat

**POS:**

- NNP
- VBD
- IN
- DT
- NN

**CHUNK:**

- NP
- VP
- PP
- NP
- NP-I

**NER:**

- PER
- -
- -
- -
- -

**SRL:**

- ARG1
- REL
- ARG2
- ARG2-I
- ARG2-I

---

**Diagram:**

- Joint training
- Blah Blah Blah
- Embedding
- Local features
  - Global features
    - Task 1 Tags
    - Task 2 Tags
    - Task 3 Tags
- Local features
  - Global features
- Local features
  - Global features
- Local features
  - Global features

**Examples:**

- The *cat* sat on the mat
- The *feline* sat on the mat

**Lookup Table:**

- Word: *cat*
- Lookup Table: $L_W$
- RED

- Word: *feline*
- Lookup Table: $L_W$
- GREEN
Words into Vectors

Idea
- Words are embed in a vector space

- Embeddings are trained

Implementation
- A word $w$ is an index in a dictionary $\mathcal{D} \in \mathbb{N}$
- Use a lookup-table ($W \sim$ feature size $\times$ dictionary size)

$$LT_W(w) = W \cdot w$$

Remarks
- Applicable to any discrete feature (words, caps, stems...)
- See (Bengio et al, 2001)
The Lookup Tables

Each word/element in dictionary maps to a vector in $\mathbb{R}^d$.

- We learn these vectors.
- LookupTable: input of $i^{th}$ word is

$$x = (0, 0, \ldots, 1, 0, \ldots, 0) \quad 1 \text{ at position } i$$

In the original space words are orthogonal.

$$\text{cat} = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, \ldots)$$
$$\text{kitten} = (0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, \ldots)$$

To get the $\mathbb{R}^d$ embedding vector for the word we multiply $Wx$ where $W$ is a $d \times N$ vector with $N$ words in the dictionary.
Window Approach

- Tags **one word** at the time
- Feed a **fixed-size window** of text around each word to tag
- **Works** fine for most tasks
- How do deal with **long-range dependencies**?
  
  *E.g.* in **SRL**, the verb of interest might be **outside the window**!
Sentence Approach

- Feed the **whole sentence** to the network
- Tag **one word** at the time: add extra **position** features
- **Convolutions** to handle variable-length inputs

![Diagram](image)

See (Bottou, 1989) or (LeCun, 1989).

- Produces **local** features with higher level of abstraction
- **Max over time** to capture most relevant features

Outputs a **fixed-sized** feature vector
Sentence Approach (2/2)

Input Sentence

Text

Feature 1

... 

Feature K

Padding

The cat sat on the mat

$w_1^1 \, w_1^2 \, ... \, w_K^1$

Padding

$w_1^K \, w_2^K \, ... \, w_K^K$

Lookup Table

$LT_{W_1}$

...

$LT_{W_K}$

Convolution

Max Over Time

$\text{max}(\cdot)$

Linear

$M^2 \times \odot$

HardTanh

Linear

$M^3 \times \odot$

$n_{K_u}$

$n_{K_u}$

$n_{K_u} = \phi$
Deep SRL

This is the network for a single window. We train/test predicting the entire sentence of tags ("structured outputs") using viterbi approach, similar to other NLP methods.
Removing The Time Dimension (1/2)
Removing The Time Dimension (2/2)
Word Tag Likelihood (WTL)

- The network has one output $f(x, i, \theta)$ per tag $i$
- Interpreted as a probability with a softmax over all tags

$$p(i | x, \theta) = \frac{e^{f(x, i, \theta)}}{\sum_j e^{f(x, j, \theta)}}$$

we can train directly for that (word tag likelihood) or we could train in a structured way by predicting the entire sentence’s tags.

That should be useful because tags are not independent.
Sentence Tag Likelihood (STL)

- The network score for tag $k$ at the $t^{th}$ word is $f(\mathbf{x}_1^T, k, t, \theta)$
- $A_{kl}$ transition score to jump from tag $k$ to tag $l$

Sentence score for a tag path $[\hat{i}]_1^T$

$$s(\mathbf{x}_1^T, [\hat{i}]_1^T, \hat{\theta}) = \sum_{t=1}^{T} \left( A_{[\hat{i}]_{t-1}[\hat{i}]_t} + f(\mathbf{x}_1^T, [\hat{i}]_t, t, \theta) \right)$$
## Supervised Benchmark Results

- **Network architectures:**
  - Window (5) approach for POS, CHUNK & NER (300HU)
  - Convolutional (3) for SRL (300+500HU)
  - Word Tag Likelihood (WTL) and Sentence Tag Likelihood (STL)

- **Network features:** lower case words (size 50), capital letters (size 5) dictionary size 100,000 words

<table>
<thead>
<tr>
<th>Approach</th>
<th>POS (PWA)</th>
<th>Chunking (F1)</th>
<th>NER (F1)</th>
<th>SRL (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark Systems</td>
<td>97.24</td>
<td>94.29</td>
<td>89.31</td>
<td>77.92</td>
</tr>
<tr>
<td>NN+WTL</td>
<td>96.31</td>
<td>89.13</td>
<td>79.53</td>
<td>55.40</td>
</tr>
<tr>
<td>NN+STL</td>
<td>96.37</td>
<td>90.33</td>
<td>81.47</td>
<td>70.99</td>
</tr>
</tbody>
</table>

- **STL helps, but... fair performance.**
- **Capacity mainly in words features... are we training it right?**
Supervised Word Embeddings

- Sentences with **similar words** should be **tagged in the same way**:
  - The **cat** sat on the mat
  - The **feline** sat on the mat

<table>
<thead>
<tr>
<th>france 454</th>
<th>jesus 1973</th>
<th>xbox 6909</th>
<th>reddish 11724</th>
<th>scratched 29869</th>
<th>megabits 87025</th>
</tr>
</thead>
<tbody>
<tr>
<td>persuade</td>
<td>thickets</td>
<td>decadent</td>
<td>widescreen</td>
<td>odd</td>
<td>ppa</td>
</tr>
<tr>
<td>faw</td>
<td>savory</td>
<td>divo</td>
<td>antica</td>
<td>anchiesta</td>
<td>uddin</td>
</tr>
<tr>
<td>blackstock</td>
<td>sympathetic</td>
<td>verus</td>
<td>shabby</td>
<td>emigration</td>
<td>biologically</td>
</tr>
<tr>
<td>giorgi</td>
<td>jfk</td>
<td>oxide</td>
<td>awe</td>
<td>marking</td>
<td>kayak</td>
</tr>
<tr>
<td>shaheed</td>
<td>khwarazm</td>
<td>urbina</td>
<td>thud</td>
<td>heuer</td>
<td>mclarens</td>
</tr>
<tr>
<td>rumelia</td>
<td>stationery</td>
<td>epos</td>
<td>occupant</td>
<td>sambhaji</td>
<td>gladwin</td>
</tr>
<tr>
<td>planum</td>
<td>ilias</td>
<td>eglinton</td>
<td>revised</td>
<td>worshippers</td>
<td>centrally</td>
</tr>
<tr>
<td>goa’uld</td>
<td>gsNUMBER</td>
<td>edging</td>
<td>leavened</td>
<td>ritsuko</td>
<td>indonesia</td>
</tr>
<tr>
<td>collation</td>
<td>operator</td>
<td>frg</td>
<td>pandionidae</td>
<td>lifeless</td>
<td>moneo</td>
</tr>
<tr>
<td>bacha</td>
<td>w.j.</td>
<td>namsos</td>
<td>shirt</td>
<td>mahan</td>
<td>nilgiris</td>
</tr>
</tbody>
</table>

- About **1M** of words in WSJ
- **15%** of most frequent words in the dictionary are seen **90%** of the time
- **Cannot expect words to be trained properly!**
Improving Word Embedding

- Rare words are not trained properly
- Sentences with similar words should be tagged in the same way:
  - The cat sat on the mat
  - The feline sat on the mat

Only 1M WSJ not enough – let’s use lots of unsupervised data!
Semi-supervised: MTL with Unlabeled Text

- Language Model: “is a sentence actually English or not?”
  Implicitly captures: * syntax * semantics
- Bengio & Ducharme (2001) Probability of next word given previous words. Overcomplicated – we do not need probabilities here
- English sentence windows: Wikipedia (~ 631M words)
  Non-english sentence windows: middle word randomly replaced
  the champion federer wins wimbledon
  vs. the champion saucepan wins wimbledon
- Multi-class margin cost:

  \[ \sum_{s \in S} \sum_{w \in D} \max (0, 1 - f(s, w^*_s) + f(s, w)) \]

  \( S \): sentence windows  \( D \): dictionary
  \( w^*_s \): true middle word in \( s \)
  \( f(s, w) \): network score for sentence \( s \) and middle word \( w \)
## Language Model: Embedding

Nearest neighbors in 100-dim. embedding space:

<table>
<thead>
<tr>
<th>FRANCE</th>
<th>JESUS</th>
<th>XBOX</th>
<th>REDDISH</th>
<th>SCRATCHED</th>
</tr>
</thead>
<tbody>
<tr>
<td>454</td>
<td>1973</td>
<td>6909</td>
<td>11724</td>
<td>29869</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPAIN</th>
<th>CHRIST</th>
<th>PLAYSTATION</th>
<th>YELLOWISH</th>
<th>SMASHED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1973</td>
<td>GOD</td>
<td>DREAMCAST</td>
<td>GREENISH</td>
<td>RIPPED</td>
</tr>
<tr>
<td>RESURRECTION</td>
<td>PSNUMBER</td>
<td>SNES</td>
<td>BROWNISH</td>
<td>RIPPPED</td>
</tr>
<tr>
<td>PRAYER</td>
<td>GOD</td>
<td>DREAMCAST</td>
<td>BLUISH</td>
<td>BRUSHED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>PSNUMBER</td>
<td>CREAMY</td>
<td>HURLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>SNES</td>
<td>WHITISH</td>
<td>GRABBED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>NINTENDO</td>
<td>BLACKISH</td>
<td>TOSSSED</td>
</tr>
<tr>
<td>RESURRECTION</td>
<td>CHRIST</td>
<td>GAMECUBE</td>
<td>SILVERY</td>
<td>SQUEEZED</td>
</tr>
<tr>
<td>PRAYER</td>
<td>RESURRECTION</td>
<td>SNES</td>
<td>GREYISH</td>
<td>BLASTED</td>
</tr>
<tr>
<td>PRAYER</td>
<td>CHRIST</td>
<td>WII</td>
<td>PALER</td>
<td>TANGLED</td>
</tr>
<tr>
<td>RESURRECTION</td>
<td>PRAYER</td>
<td>NES</td>
<td>PALER</td>
<td>SLASHED</td>
</tr>
<tr>
<td>RESURRECTION</td>
<td>CHRIST</td>
<td>NES</td>
<td>PALER</td>
<td>SLASHED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>NES</td>
<td>PALER</td>
<td>SLASHED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>SNES</td>
<td>PALER</td>
<td>SLASHED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>NINTENDO</td>
<td>BLACKISH</td>
<td>SQUEEZED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>GAMECUBE</td>
<td>SILVERY</td>
<td>BLAGHTED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>SNES</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>NINTENDO</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>GAMECUBE</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>SNES</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>NINTENDO</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>GAMECUBE</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>CHRIST</td>
<td>SNES</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>CHRIST</td>
<td>RESURRECTION</td>
<td>NINTENDO</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
</tbody>
</table>

(Even fairly rare words are embedded well.)
## Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>POS (PWA)</th>
<th>CHUNK (F1)</th>
<th>NER (F1)</th>
<th>SRL (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td>97.24</td>
<td>94.29</td>
<td>89.31</td>
<td>77.92</td>
</tr>
<tr>
<td>[Toutanova ’03]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Sha ’03]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Ando ’05]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Koomen ’05]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NN + WTL</strong></td>
<td>96.31</td>
<td>89.13</td>
<td>79.53</td>
<td>55.40</td>
</tr>
<tr>
<td><strong>NN + STL</strong></td>
<td>96.37</td>
<td>90.33</td>
<td>81.47</td>
<td>70.99</td>
</tr>
<tr>
<td><strong>NN + LM + STL</strong></td>
<td>97.22</td>
<td>94.10</td>
<td>88.67</td>
<td>74.15</td>
</tr>
<tr>
<td><strong>NN + . . . + tricks</strong></td>
<td>97.29</td>
<td>94.32</td>
<td>89.95</td>
<td>76.03</td>
</tr>
<tr>
<td>[+suffix]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[+POS]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[+gazetteer]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[+Parse Trees]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NOTES:
- Didn’t compare to benchmarks that used external labeled data.
- [Ando ’05] uses external unlabeled data.
- [Koomen ’05] uses 4 parse trees not provided by the challenge. Using only 1 tree it gets 74.76.
## Software

Code for tagging with POS, NER, CHUNK, SRL + parse trees:

http://ml.nec-labs.com/senna/

<table>
<thead>
<tr>
<th>System</th>
<th>RAM (Mb)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toutanova, 2003</td>
<td>1100</td>
<td>1065</td>
</tr>
<tr>
<td>Shen, 2007</td>
<td>2200</td>
<td>833</td>
</tr>
<tr>
<td>SENNA</td>
<td>32</td>
<td>4</td>
</tr>
</tbody>
</table>

(a) POS

<table>
<thead>
<tr>
<th>System</th>
<th>RAM (Mb)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koomen, 2005</td>
<td>3400</td>
<td>6253</td>
</tr>
<tr>
<td>SENNA</td>
<td>124</td>
<td>52</td>
</tr>
</tbody>
</table>

(b) SRL

See also Torch: http://www.torch.ch
Recursive NNs for Parsing, Sentiment, ... and more!
(Socher et al., ICML ’13), (Socher et al., EMNLP, ’13))

Build sentence representations using the parse tree to compose embeddings via a nonlinear function taking pairs $(c_1, c_2)$ and output $p$.

\[
s = W^{\text{score}} p \quad (9)
\]
\[
p = f(W[c_1; c_2] + b)
\]
Paragraph Vector

(Le & Mikolov, ’14)

A Paragraph Vector (a vector that represents a paragraph/doc) learned by:

1) Predicting the words in a doc; 2) predict $n$-grams in the doc:

At test time, for a new document, one needs to learn its vector, this can encode word order via the $n$-gram prediction approach.
# Comparison of CNN, RNN & PV (Kim '14)

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>SST-1</th>
<th>SST-2</th>
<th>Subj</th>
<th>TREC</th>
<th>CR</th>
<th>MPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-rand</td>
<td>76.1</td>
<td>45.0</td>
<td>82.7</td>
<td>89.6</td>
<td>91.2</td>
<td>79.8</td>
<td>83.4</td>
</tr>
<tr>
<td>CNN-static</td>
<td>81.0</td>
<td>45.5</td>
<td>86.8</td>
<td>93.0</td>
<td>92.8</td>
<td>84.7</td>
<td>89.6</td>
</tr>
<tr>
<td>CNN-non-static</td>
<td>81.5</td>
<td>48.0</td>
<td>87.2</td>
<td>93.4</td>
<td>93.6</td>
<td>84.3</td>
<td>89.5</td>
</tr>
<tr>
<td>CNN-multichannel</td>
<td>81.1</td>
<td>47.4</td>
<td>88.1</td>
<td>93.2</td>
<td>92.2</td>
<td>85.0</td>
<td>89.4</td>
</tr>
<tr>
<td>RAE (Socher et al., 2011)</td>
<td>77.7</td>
<td>43.2</td>
<td>82.4</td>
<td></td>
<td></td>
<td></td>
<td>86.4</td>
</tr>
<tr>
<td>MV-RNN (Socher et al., 2012)</td>
<td>79.0</td>
<td>44.4</td>
<td>82.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNTN (Socher et al., 2013)</td>
<td></td>
<td>45.7</td>
<td>85.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCNN (Kalchbrenner et al., 2014)</td>
<td></td>
<td>48.5</td>
<td>86.8</td>
<td></td>
<td>93.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paragraph-Vec (Le and Mikolov, 2014)</td>
<td></td>
<td>48.7</td>
<td>87.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAE (Hermann and Blunsom, 2013)</td>
<td>77.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>87.2</td>
</tr>
<tr>
<td>Sent-Parser (Dong et al., 2014)</td>
<td>79.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>86.3</td>
</tr>
<tr>
<td>NBSVM (Wang and Manning, 2012)</td>
<td>79.4</td>
<td></td>
<td></td>
<td>93.2</td>
<td></td>
<td>81.8</td>
<td>86.3</td>
</tr>
<tr>
<td>MNB (Wang and Manning, 2012)</td>
<td>79.0</td>
<td></td>
<td></td>
<td></td>
<td>93.6</td>
<td>80.0</td>
<td>86.3</td>
</tr>
<tr>
<td>G-Dropout (Wang and Manning, 2013)</td>
<td>79.0</td>
<td></td>
<td></td>
<td>93.4</td>
<td></td>
<td>82.1</td>
<td>86.1</td>
</tr>
<tr>
<td>F-Dropout (Wang and Manning, 2013)</td>
<td>79.1</td>
<td></td>
<td></td>
<td></td>
<td>93.6</td>
<td>81.9</td>
<td>86.3</td>
</tr>
<tr>
<td>Tree-CRF (Nakagawa et al., 2010)</td>
<td>77.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81.4</td>
<td>86.1</td>
</tr>
<tr>
<td>CRF-PR (Yang and Cardie, 2014)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82.7</td>
<td></td>
</tr>
<tr>
<td>SVM$_S$ (Silva et al., 2011)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some More Recent Work

- **Compositionality approaches by Marco Baroni’s group:**
  > Words are combined with linear matrices dependent on the P.O.S.: G. Dinu and M. Baroni. How to make words with vectors: Phrase generation in distributional semantics. ACL ’14.

- **Document representation by Phil Blunson’s group:**
  > Variants of convolutional networks for text:
  > Kalchbrenner et al. A Convolutional Neural Network for Modelling Sentences. ACL ’14

Good tutorial slides from these teams covering multiple topics:

New Directions in Vector Space Models of Meaning
http://www.cs.ox.ac.uk/files/6605/aclVectorTutorial.pdf
Summary

- Generic end-to-end deep learning system for NLP tasks.

- Word embeddings combined to form sentence or document embeddings can perform well on supervised tasks.

- Previous common belief in NLP: engineering syntactic features necessary for semantic tasks.
  One can do well by engineering a model/algorithm rather than features.

Attitude is changing in recent years... let’s see what happens!
Embedding Methods for NLP

Part 2: Embeddings for Multi-relational Data

Antoine Bordes & Jason Weston
Facebook AI Research

EMNLP tutorial – October 29, 2014
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2. Pros and cons of embedding models

3. Future of embedding models

4. Resources
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2. Pros and cons of embedding models

3. Future of embedding models

4. Resources
Multi-relational data

- Data is structured as a graph
- Each node = an entity
- Each edge = a relation/fact
- A relation = \((sub, rel, obj)\):
  - \(sub =\) subject,
  - \(rel =\) relation type,
  - \(obj =\) object.
- Nodes w/o features.

In this talk, we focus on Knowledge Bases (KBs).
Example of KB: WordNet

- **WordNet**: dictionary where each entity is a sense (synset).

- Popular in NLP.
- **Statistics**:
  - 117k entities;
  - 20 relation types;
  - 500k facts.
- **Examples**:
  (car\_NN\_1, \_has\_part, \_wheel\_NN\_1)
  (score\_NN\_1, \_is\_a, \_rating\_NN\_1)
  (score\_NN\_2, \_is\_a, \_sheet\_music\_NN\_1)
Example of KB: Freebase

**Freebase**: huge collaborative (hence noisy) KB.

Part of the Google Knowledge Graph.

Statistics:
- 80M of entities;
- 20k relation types;
- 1.2B facts.

Examples:
(Barack Obama, _place_of_birth, Hawai)
(Albert Einstein, _follows_diet, Veganism)
(San Francisco, _contains, Telegraph Hill)
Modeling Knowledge Bases

Why KBs?
- **KBs**: Semantic search, connect people and things
- **KBs ↔ Text**: Information extraction
- **KBs → Text**: Text interpretation, summary, Q&A

Main issue: KBs are hard to manipulate
- Large dimensions: $10^5/10^8$ entities, $10^4/10^6$ rel. types
- Sparse: few valid links
- Noisy/incomplete: missing/wrong relations/entities

How?
1. Encode KBs into low-dimensional vector spaces
2. Use these representations:
   - to complete/visualize KBs
   - as KB data in text applications
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2. Pros and cons of embedding models

3. Future of embedding models

4. Resources
Link Prediction

Add new facts **without requiring extra knowledge**

From known information, **assess the validity of an unknown fact**
Link Prediction

Add new facts without requiring extra knowledge

From known information, assess the validity of an unknown fact

→ collective classification
→ reasoning in embedding spaces
Statistical Relational Learning

- **Framework:**
  - $n_s$ subjects $\{sub_i\}_{i \in [1;n_s]}$
  - $n_r$ relation types $\{rel_k\}_{k \in [1;n_r]}$
  - $n_o$ objects $\{obj_j\}_{j \in [1;n_o]}$
  - For us, $n_s = n_o = n_e$ and $\forall i \in [1;n_e], sub_i = obj_i$.
  - A fact exists for $(sub_i, rel_k, obj_j)$ if $rel_k(sub_i, obj_j) = 1$

- **Goal:** We want to model, from data,

$$P[rel_k(sub_i, obj_j) = 1]$$

(eq. to approximate the binary tensor $\mathbf{X} \in \{0, 1\}^{n_s \times n_o \times n_r}$)
Previous Work

- Tensor factorization (Harshman et al., '94)
- Probabilistic Relational Learning (Friedman et al., '99)
- Relational Markov Networks (Taskar et al., '02)
- Markov-logic Networks (Kok et al., '07)
- Extension of SBMs (Kemp et al., '06) (Sutskever et al., '10)
- Spectral clustering (undirected graphs) (Dong et al., '12)
- Ranking of random walks (Lao et al., '11)
- Collective matrix factorization (Nickel et al., '11)
- Embedding models (Bordes et al., '11, '13) (Jenatton et al., '12)
  (Socher et al., '13) (Wang et al., '14) (García-Durán et al., '14)
Collective Matrix Factorization (Nickel et al., '11)

- **RESCAL**: $\forall k \in [1; n_r], R_k \in \mathbb{R}^{d \times d}$ and $A \in \mathbb{R}^{ne \times d}$ (close from DEDICOM (Harshman, '78)).

- $A$ & $R$ learned by reconstruction (alternating least-squares):

$$
\min_{A,R} \frac{1}{2} \left( \sum_k \| X_k - A R_k A^T \|_F^2 \right) + \lambda_A \| A \|_F^2 + \lambda_R \sum_k \| R_k \|_F^2
$$
### Scalability

<table>
<thead>
<tr>
<th>Method</th>
<th>Nb of parameters</th>
<th>on Freebase15k</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL</td>
<td>$O(n_e d + n_r d^2)$</td>
<td>88M ($d = 250$)</td>
</tr>
</tbody>
</table>

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- RESCAL involves many parameters.
- Bad scalability w.r.t. $n_r$.
- Reconstruction criterion does not fit well for binary data.
Embedding Models

Two main ideas:

1. Models based on low-dimensional continuous vector embeddings for entities and relation types, directly trained to define a similarity criterion.

2. Stochastic training based on ranking loss with sub-sampling of unknown relations.
Embedding Models for KBs

- Subjects and objects are represented by vectors in $\mathbb{R}^d$.
  - $\{\text{sub}_i\}_{i \in [1; n_s]} \rightarrow [s^1, \ldots, s^{n_s}] \in \mathbb{R}^{d \times n_s}$
  - $\{\text{obj}_j\}_{j \in [1; n_o]} \rightarrow [o^1, \ldots, o^{n_o}] \in \mathbb{R}^{d \times n_o}$
  For us, $n_s = n_o = n_e$ and $\forall i \in [1; n_e], s_i = o_i$.

- Rel. types = similarity operators between subj/obj.
  - $\{\text{rel}_k\}_{k \in [1; n_r]} \rightarrow$ operators $\{R_k\}_{k \in [1; n_r]}$

- Learning similarities depending on $rel \rightarrow d(sub, rel, obj)$, parameterized by $s$, $R$ and $o$. 
Structured Embeddings (Bordes et al., ’11)

Intuition: \( \text{sub} \) and \( \text{obj} \) are projected using \( \text{rel} \) in a space where they are similar

\[
d(\text{sub}, \text{rel}, \text{obj}) = -\| \text{R}^{\text{left}} \text{s}^\top - \text{R}^{\text{right}} \text{o}^\top \|_1
\]

- Entities: \( \text{s} \) and \( \text{o} \) ∈ \( \mathbb{R}^d \)
- Projection: \( \text{R}^{\text{left}} \) and \( \text{R}^{\text{right}} \) ∈ \( \mathbb{R}^{d \times d} \)
  \( \text{R}^{\text{left}} \neq \text{R}^{\text{right}} \) because of asymmetry
- Similarity: \( \text{L1} \) distance
Stochastic Training

- Learning by **stochastic gradient descent**: one training fact after the other

- For each relation from the training set:
  1. sub-sample unobserved facts (false?)
  2. check if the similarity of the true fact is lower
  3. if **not**, update parameters of the considered facts

- Stopping criterion: performance on a validation set
Scalability

<table>
<thead>
<tr>
<th>Method</th>
<th>Nb of parameters</th>
<th>on Freebase15k</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL</td>
<td>$O(n_e d + n_r d^2)$</td>
<td>88M ($d = 250$)</td>
</tr>
<tr>
<td>SE</td>
<td>$O(n_e d + 2n_r d^2)$</td>
<td>8M ($d = 50$)</td>
</tr>
</tbody>
</table>

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- SE also involves many parameters.
- Bad scalability w.r.t. $n_r$.
- Potential training problems for SE (overfitting).
Neural Tensor Networks (Socher et al., ’13)

- In NTN, a relationship is represented by a tensor, 2 matrices and 2 vectors + a non-linearity ($tanh$).

$$d(sub, rel, obj) = u_r^T \tanh (h^T W_r t + V^1_r h + V^2_r t + b_r)$$

- Neural Tensor layer:

- Very powerful model with high capacity for each relation.
## Scalability

<table>
<thead>
<tr>
<th>Method</th>
<th>Nb of parameters</th>
<th>on Freebase15k</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL</td>
<td>$O(n_e d + n_r d^2)$</td>
<td>88M ($d = 250$)</td>
</tr>
<tr>
<td>SE</td>
<td>$O(n_e d + 2n_r d^2)$</td>
<td>8M ($d = 50$)</td>
</tr>
<tr>
<td>NTN</td>
<td>$O(n_e d + n_r d^3)$</td>
<td>165M ($d = 50$)</td>
</tr>
</tbody>
</table>

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- Very high modeling capacity.
- Involves many parameters.
- Bad scalability w.r.t. $n_r$ (overfitting if few triples).
Intuition: we want $s + r \approx o$. 

Modeling Relations as Translations (Bordes et al. ’13)
Modeling Relations as Translations (NIPS13)

**Intuition:** we would like that $s + r \approx o$. 

![Diagram showing relations between entities like Austin, Miami, John, Patti, Jane, and their relationships such as mom and child_of.](image-url)
Modeling Relations as Translations (Bordes et al. ’13)

**Intuition:** we want \( s + r \approx o \).

The similarity measure is defined as:

\[
d(sub, rel, obj) = \|s + r - o\|^2_2
\]

\( s, r \) and \( o \) are learned to verify that.
Learning TransE

For training, a margin ranking criterion is minimized:

\[
\sum_{\text{pos}} \sum_{\text{neg} \in S'} \left[ \gamma + \|s + r - o\|_2^2 - \|s' + r - o'\|_2^2 \right]_+ 
\]

where \([x]_+\) is the positive part of \(x\), \(\gamma > 0\) is a margin, and:

\[
S' = \{(\text{sub}',\text{rel},\text{obj})|\text{sub}' \in \mathcal{E}\} \cup \{(\text{sub},\text{rel},\text{obj}')|\text{obj}' \in \mathcal{E}\}
\]
Learning TransE

1: **input:** Training set \( S = \{(\text{sub}, \text{rel}, \text{obj})\} \), margin \( \gamma \), learning rate \( \lambda \)
2: **initialize** \( r \leftarrow \text{uniform}( -\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}} ) \) for each rel
3: \( r \leftarrow \ell / \| \ell \| \) for each \( \ell \)
4: \( e \leftarrow \text{uniform}( -\frac{6}{\sqrt{j}}, \frac{6}{\sqrt{k}} ) \) for each entity \( \text{ent} (\text{sub or obj}) \)
5: **loop**
6: \( e \leftarrow e / \| e \| \) for each entity \( \text{ent} \)
7: \( S_{batch} \leftarrow \text{sample}(S, b) \) //sample minibatch of size \( b \)
8: \( T_{batch} \leftarrow \emptyset \) //initialize set of pairs
9: **for** \( (\text{sub}, \text{rel}, \text{obj}) \in S_{batch} \) **do**
10: \( (\text{sub}', \text{rel}, \text{obj}') \leftarrow \text{sample}(S'(\text{sub}, \text{rel}, \text{obj})) \) //sample negative triplet
11: \( T_{batch} \leftarrow T_{batch} \cup \{(\text{sub}, \text{rel}, \text{obj}), (\text{sub}', \text{rel}, \text{obj}')\} \)
12: **end for**
13: Update embeddings w.r.t. \( \sum_{T_{batch}} \nabla \left[ \gamma + \| s + r - o \|^2_2 - \| s' + r - o' \|^2_2 \right]_+ \)
14: **end loop**
Motivations of a Translation-based Model

- Natural representation for hierarchical relationships.

- Recent work on word embeddings (Mikolov et al., ’13): there may exist embedding spaces in which relationships among concepts are represented by translations.
## Scalability

<table>
<thead>
<tr>
<th>Method</th>
<th>Nb of parameters</th>
<th>on Freebase15k</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL</td>
<td>$O(n_e d + n_r d^2)$</td>
<td>88M ($d = 250$)</td>
</tr>
<tr>
<td>SE</td>
<td>$O(n_e d + 2n_r d^2)$</td>
<td>8M ($d = 50$)</td>
</tr>
<tr>
<td>NTN</td>
<td>$O(n_e d + n_r d^3)$</td>
<td>165M ($d = 50$)</td>
</tr>
<tr>
<td>TransE</td>
<td>$O(n_e d + n_r d)$</td>
<td>0.8M ($d = 50$)</td>
</tr>
</tbody>
</table>

Freebase15k: $n_e = 15k$, $n_r = 1.3k$.

- TransE is a special case of SE and NTN.
- TransE obtains better training errors: less overfitting.
- Much better scalability.
Chunks of Freebase

Data statistics:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FB13</td>
<td>75,043</td>
<td>13</td>
<td>316,232</td>
<td>5,908</td>
<td>23,733</td>
</tr>
<tr>
<td>FB15k</td>
<td>14,951</td>
<td>1,345</td>
<td>483,142</td>
<td>50,000</td>
<td>59,071</td>
</tr>
<tr>
<td>FB1M</td>
<td>$1 \times 10^6$</td>
<td>23,382</td>
<td>$17.5 \times 10^6$</td>
<td>50,000</td>
<td>177,404</td>
</tr>
</tbody>
</table>

Training times for TransE:

- Embedding dimension: 50.
- Training time:
  - on Freebase15k: $\approx2h$ (on 1 core),
  - on Freebase1M: $\approx1d$ (on 16 cores).
Visualization of 1,000 Entities
Visualization of 1,000 Entities - Zoom 1
Visualization of 1,000 Entities - Zoom 2
Visualization of 1,000 Entities - Zoom 3
Example

"Who influenced J.K. Rowling?"

J. K. Rowling _influenced_by_ ?
Example

"Who influenced J.K. Rowling?"

J. K. Rowling influenced by G. K. Chesterton
J. R. R. Tolkien
C. S. Lewis
Lloyd Alexander
Terry Pratchett
Roald Dahl
Jorge Luis Borges
Stephen King
Ian Fleming
Example

"Which genre is the movie WALL-E?"

WALL-E  _has_genre_  ?
Example

”Which genre is the movie WALL-E?”

WALL-E \_has\_genre Animation
Computer animation
Comedy film
Adventure film
Science Fiction
Fantasy
Stop motion
Satire
Drama
On **FB1M**, TransE predicts **34% in the Top-10** (SE only **17.5%**). Results extracted from (Bordes et al., ’13) and (Wang et al., ’14)
Refining TransE

- **TATEC** (García-Durán et al., ’14) supplements TransE with a **trigram term** for encoding complex relationships:

\[
d(sub, rel, obj) = s_1^\top R o_1 + s_2^\top r + o_2^\top r' + s_2^\top D o_2,
\]

with \(s_1 \neq s_2\) and \(o_1 \neq o_2\).

- **TransH** (Wang et al., ’14) adds an **orthogonal projection** to the translation of TransE:

\[
d(sub, rel, obj) = \|(s-r_p^\top s r_p) + r_t - (o-r_p^\top o r_p)\|_2^2,
\]

with \(r_p \perp r_t\).
Benchmarking

Ranking on FB15k

Results extracted from (García-Durán et al., ’14) and (Wang et al., ’14)
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2. Pros and cons of embedding models

3. Future of embedding models

4. Resources
Information Extraction

- Information extraction: populate KBs with new facts using text
- Usually two steps:
  - **Entity linking**: identify mentions of entities in text
  - **Relation extraction**: extract facts about them
- Previous works include rule-based models, classifiers with features from parsers, graphical models, etc.
- Embedding models exist for both steps.
Entity Linking as WSD

Word Sense Disambiguation ↔ WordNet entity linking

Towards open-text semantic parsing:

```
``A musical score accompanies a television program."

\[\downarrow\text{Semantic Role Labeling}\]

(``A musical score'', ``accompanies'', ``a television program'')

\[\downarrow\text{Preprocessing (POS, Chunking, ...)}\]

((\_musical\_JJ score\_NN ),\_accompany\_VB ,\_television\_program\_NN )

\[\downarrow\text{Word-sense Disambiguation}\]

((\_musical\_JJ\_1 score\_NN\_2),\_accompany\_VB\_1,\_television\_program\_NN\_1)
Embeddings of Text and WordNet (Bordes et al., ’12)

- Text is converted into relations \((sub, rel, obj)\).
- Joint learning of embeddings for all symbols: words, entities and relation types from WordNet.
- This system can label **37,141 words** with **40,943 synsets**.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Train. Ex.</th>
<th>Test Ex.</th>
<th>Labeled?</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>146,442</td>
<td>5,000</td>
<td>No</td>
<td>synsets</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>2,146,131</td>
<td>10,000</td>
<td>No</td>
<td>words</td>
</tr>
<tr>
<td>ConceptNet</td>
<td>11,332</td>
<td>0</td>
<td>Non</td>
<td>words</td>
</tr>
<tr>
<td>Ext. WordNet</td>
<td>42,957</td>
<td>5,000</td>
<td>Yes</td>
<td>words+synsets</td>
</tr>
<tr>
<td>Unamb. Wikip.</td>
<td>981,841</td>
<td>0</td>
<td>Yes</td>
<td>words+synsets</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3,328,703</td>
<td>20,000</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Benchmarking on Extended WordNet

F1-score on 5,000 test sentences to disambiguate.

Results extracted from (Bordes et al., '12)
WordNet is enriched through text

Similarities among senses beyond WordNet

"what does an army attack?"

army_NN_1  attack_VB_1  ?
WordNet is enriched through text

Similarities among senses beyond original WordNet data

”what does an army attack?”

army_NN_1  attack_VB_1  troop_NN_4
armed_service_NN_1  ship_NN_1
territory_NN_1  military_unit_NN_1
WordNet is enriched through text

Similarities among senses beyond WordNet

”Who or what earns money”

? earn_VB_1 money_NN_1
WordNet is enriched through text

Similarities among senses beyond original WordNet data

"Who or what earns money"

person_NN_1 earn_VB_1 money_NN_1
business_firm_NN_1
family_NN_1
payoff_NN_3
card_game_NN_1
Relation Extraction

Given a bunch of sentences.

Text:  "Alfred Hitchcock, who wrote and directed, The Birds"  
"M. Hitchcock, on the set of the movie The Birds"  
"Sir A. Hitchcock, the famous director of The Birds"
Relation Extraction

Given a bunch of sentences concerning the same pair of entities.

Freebase: /m/2d3rf

Text: "Alfred Hitchcock", who wrote and directed, on the set of, the famous director of "The Birds", the movie "The Birds"
Relation Extraction

**Goal:** identify if there is a relation between them to add to the KB.
Relation Extraction

And from which type, to enrich an existing KB.

Freebase: /m/2d3rf

Text:
"Alfred Hitchcock" "M. Hitchcock" "Sir A. Hitchcock"

"who wrote and directed, on the set of, the famous director of"

The Birds" the movie The Birds"

The Birds"
Embeddings of Text and Freebase (Weston et al., ’13)

- **Standard Method:** an embedding-based classifier is trained to predict the relation type, given text mentions $\mathcal{M}$ and $(\text{sub}, \text{obj})$:

$$r(m, \text{sub}, \text{obj}) = \arg \max_{rel'} \sum_{m \in \mathcal{M}} S_{m2r}(m, rel')$$

Classifier based on WSABIE (Weston et al., ’11).
Embeddings of Text and Freebase (Weston et al., ’13)

- **Idea:** improve extraction by using both text + available knowledge (= current KB).

- A model of the KB is used in a re-ranking setting to force extracted relations to agree with it:

  \[
  r'(m, sub, obj) = \arg \max_{rel'} \left( \sum_{m \in M} S_{m2r}(m, rel') - d_{KB}(sub, rel', obj) \right)
  \]

  with \(d_{KB}(sub, rel', obj) = \|s + r' - o\|^2_2\) (trained separately)
Benchmarking on NYT+Freebase

Exp. on NY Times papers linked with Freebase (Riedel et al., '10)

Precision/recall curve for predicting relations

Results extracted from (Weston et al., '13)
Universal Schemas (Riedel et al., ’13)

- Join in a single learning problem:
  - relation extraction
  - link prediction
- The same model score triples:
  - made of text mentions
  - from a KB
Universal Schemas  (Riedel et al., ’13)

- Relation prediction using the score:

$$ r'(m, sub, obj) = \arg \max_{rel'} \left( \sum_{m \in M} S_{m2r}(m, rel') + S_{KB}(sub, rel', obj) + S_{neighbors}(sub, rel', obj) \right) $$

- All scores are defined using embeddings:
  - $$ S_{m2r}(m, rel') = f(m)^\top r' $$
  - $$ S_{kb}(sub, rel', obj) = s^\top r'_s + o^\top r'_o $$
  - $$ S_{neighbors}(sub, rel', obj) = \sum_{(sub, rel'', obj) \text{ such that } rel'' \neq rel'} w_{rel''}^{rel'} $$

- Training by ranking observed facts versus other and updating using SGD.
Benchmarking on NYT+Freebase

Exp. on NY Times papers linked with Freebase (Riedel et al., ’10)

Mean Averaged Precision for predicting relations

Results extracted from (Riedel et al., ’13)
Menu – Part 2

1 Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2 Pros and cons of embedding models

3 Future of embedding models

4 Resources
Link Prediction as Q&A

”Who influenced J.K. Rowling?”

J. K. Rowling  _influenced_by  G. K. Chesterton
J. R. R. Tolkien
C. S. Lewis
Lloyd Alexander
Terry Pratchett
Roald Dahl
Jorge Luis Borges

Can we go beyond such rigid structure?
Open-domain Question Answering

- **Open-domain Q&A**: answer question on any topic
  \[\rightarrow\] query a KB with natural language

**Examples**

- “What is cher’s son’s name?”  
  Elijah_blue_allman
- “What are dollars called in spain?”  
  Peseta
- “What is henry_clay known for?”  
  Lawyer
- “Who did georges_clooney marry in 1987?”  
  Kelly_preston

- Recent effort with **semantic parsing** (Kwiatkowski et al. ’13)  
  (Berant et al. ’13, ’14) (Fader et al., ’13, ’14) (Reddy et al., ’14)
- Models with **embeddings** as well (Bordes et al., ’14)
Subgraph Embeddings *(Bordes et al., ’14)*

- Model learns *embeddings of questions and (candidate) answers*
- Answers are represented by entity and its *neighboring subgraph*

---

**Question**

“Who did Clooney marry in 1987?”

**Freebase subgraph**

- G. Clooney
- K. Preston
- Honolulu
- 1987
- J. Travolta

**Embedding model**

- Embedding of the question
- Word embedding lookup table
- Binary encoding of the question

**Freebase embedding lookup table**

**Score**

How the candidate answer fits the question

**Dot product**

Embedding of the subgraph

**Binary encoding of the subgraph**

**Subgraph of a candidate answer (here K. Preston)**

Model
Training data

- Freebase is automatically converted into Q&A pairs
- Closer to expected language structure than triples

Examples of Freebase data

(sikkim, location.in_state.judicial_capital, gangtok)
what is the judicial capital of the in state sikkim ? – gangtok

(brighouse, location.location.people_born_here, edward_barber)
who is born in the location brighouse ? – edward_barber

(sepsis, medicine.disease.symptoms, skin_discoloration)
what are the symptoms of the disease sepsis ? – skin_discoloration
Training data

- All Freebase questions have rigid and similar structures
- Supplemented by pairs from clusters of paraphrase questions
- Multitask training: similar questions $\leftrightarrow$ similar embeddings

Examples of paraphrase clusters

- what are two reasons to get a 404?
- what is error 404?
- how do you correct error 404?

- what is the term for a teacher of Islamic law?
- what is the name of the religious book Islam uses?
- who is the chief of Islamic religious authority?

- what country is Bueno Aire in?
- what country is Buenos Aires in?
- what country is Bueno are in?
Benchmarking on WebQuestions

Experiments on WebQuestions (Berant et al., ’13)

F1-score for answering test questions

Results extracted from (Berant et al., ’14) and (Bordes et al., ’14)
Menu – Part 2

1 Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2 Pros and cons of embedding models

3 Future of embedding models

4 Resources
Advantages

- **Efficient features** for many tasks in practice
- Training with SGD **scales & parallelizable** (Niu et al., ’11)
- Flexible to various tasks: **multi-task learning of embeddings**
- **Supervised or unsupervised training**
- Allow to use **extra-knowledge in other applications**
Issues

- Must train all embeddings together (no parallel 1-vs-rest)
- Low-dimensional vector $\rightarrow$ compression, blurring
- Sequential models suffer from long-term memory
- Embeddings need quite some updates to be good – not 1-shot
- Negative example sampling can be inefficient
Menu – Part 2

1. Embeddings for multi-relational data
   - Multi-relational data
   - Link Prediction in KBs
   - Embeddings for information extraction
   - Question Answering

2. Pros and cons of embedding models

3. Future of embedding models

4. Resources
Fix current limitations

- **Compression**: improve the memory capacity of embeddings and allows for one-shot learning of new symbols
- **Long-term memory**: encode longer dependencies in sequential models like RNNs
- **Training**: faster and better sampling of examples
- **Beyond linear**: most supervised labeling problems are well tackled by simple linear models. Non-linearity should help more.
Explore new directions

- **Compositionality** (Baroni et al. ’10) (Grefenstette, 13)
- **Multimodality** (Bruni et al., 12) (Kiros et al., ’14)
- **Grounding language into actions** (Bordes et al., 10)
At EMNLP

Modeling Interestingness with Deep Neural Networks
Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He and Li Deng

Translation Modeling with Bidirectional Recurrent Neural Networks
Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker and Hermann Ney

Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics
Douwe Kiela and Léon Bottou

Learning Abstract Concept Embeddings from Multi-Modal Data: Since You Probably Can’t See What I Mean
Felix Hill and Anna Korhonen

Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases
Matt Gardner, Partha Talukdar, Jayant Krishnamurthy and Tom Mitchell

Composition of Word Representations Improves Semantic Role Labelling
Michael Roth and Kristian Woodsend
At EMNLP

A Neural Network for Factoid Question Answering over Paragraphs
Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher and Hal Daumé III

Joint Relational Embeddings for Knowledge-based Question Answering
Min-Chul Yang, Nan Duan, Ming Zhou and Hae-Chang Rim

Evaluating Neural Word Representations in Tensor-Based Compositional Settings
Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh and Matthew Purver

Opinion Mining with Deep Recurrent Neural Networks
Ozan Irsoy and Claire Cardie

The Inside-Outside Recursive Neural Network model for Dependency Parsing
Phong Le and Willem Zuidema

A Fast and Accurate Dependency Parser using Neural Networks
Danqi Chen and Christopher Manning
At EMNLP

Reducing Dimensions of Tensors in Type-Driven Distributional Semantics
Tamara Polajnar, Luana Fagarasan and Stephen Clark

Word Semantic Representations using Bayesian Probabilistic Tensor Factorization
Jingwei Zhang, Jeremy Salwen, Michael Glass and Alfio Gliozzo

Glove: Global Vectors for Word Representation
Jeffrey Pennington, Richard Socher and Christopher Manning

Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures
Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa and Yoshimasa Tsuruoka

Typed Tensor Decomposition of Knowledge Bases for Relation Extraction
Kai-Wei Chang, Wen-tau Yih, Bishan Yang and Christopher Meek

Knowledge Graph and Text Jointly Embedding
Zhen Wang, Jianwen Zhang, Jianlin Feng and Zheng Chen
At EMNLP

Question Answering with Subgraph Embeddings
Antoine Bordes, Sumit Chopra and Jason Weston

Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks
Ke M. Tran, Arianna Bisazza and Christof Monz

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio

Convolutional Neural Networks for Sentence Classification
Yoon Kim

#TagSpace: Semantic Embeddings from Hashtags
Jason Weston, Sumit Chopra and Keith Adams
Menu – Part 2

1 Embeddings for multi-relational data
   • Multi-relational data
   • Link Prediction in KBs
   • Embeddings for information extraction
   • Question Answering

2 Pros and cons of embedding models

3 Future of embedding models

4 Resources
Code

- **Torch**: [www.torch.ch](http://www.torch.ch)
- **SENNA**: [ronan.collobert.com/senna](http://ronan.collobert.com/senna)
- **RNNLM**: [www.fit.vutbr.cz/~imikolov/rnnlm](http://www.fit.vutbr.cz/~imikolov/rnnlm)
- **Word2vec**: [code.google.com/p/word2vec](http://code.google.com/p/word2vec)
- **Recursive NN**: [nlp.stanford.edu/sentiment](http://nlp.stanford.edu/sentiment)
- **SME (multi-relational data)**: [github.com/glorotxa/sme](http://github.com/glorotxa/sme)
References

Semantic Parsing on Freebase from Question-Answer Pairs
J. Berant, A. Chou, R. Frostig & P. Liang. *EMNLP, 2013*

Semantic Parsing via Paraphrasing
J. Berant & P. Liang. *ACL, 2013*

Learning Structured Embeddings of Knowledge Bases
A. Bordes, J. Weston, R. Collobert & Y. Bengio. *AAAI, 2011*

Joint Learning of Words and Meaning Rep. for Open-Text Semantic Parsing
A. Bordes, X. Glorot, J. Weston & Y. Bengio. *AISTATS, 2012*

A Semantic Matching Energy Function for Learning with Multi-relational Data
A. Bordes, X. Glorot, J. Weston & Y. Bengio. *MLJ, 2013*

Translating Embeddings for Modeling Multi-relational Data
References

Question Answering with Subgraph Embeddings
A. Bordes, S. Chopra & J. Weston. *EMNLP*, 2014

Clustering with Multi-Layer Graphs: A Spectral Perspective

Paraphrase-Driven Learning for Open Question Answering

Open Question Answering Over Curated and Extracted Knowledge Bases

Learning Probabilistic Relational Models
N. Friedman, L. Getoor, D. Koller & A. Pfeffer. *IJCAI*, 1999

Effective Blending of Two and Three-way Interactions for Modeling Multi-relational Data
References

Models for the Analysis of Asymmetrical Relationships among n Objects or Stimuli

PARAFAC: Parallel factor analysis
R. Harshman & M. Lundy. *Comp. Statistics and Data Analysis, 1994*

A Latent Factor Model for Highly Multi-relational Data

Learning Systems of Concepts with an Infinite Relational Model

Statistical Predicate Invention
S. Kok, P. Domingos. *ICML, 2007*

Scaling Semantic Parsers with On-the-fly Ontology Matching
References

Random Walk Inference and Learning in A Large Scale Knowledge Base

Distributed Representations of Words and Phrases and their Compositionality

A Three-Way Model for Collective Learning on Multi-Relational Data

Large-scale Semantic Parsing without Question-Answer Pairs

Modeling Relations and Their Mentions without Labeled Text
S. Riedel, L. Yao and A. McCallum. ECML-PKDD, 2010

Relation Extraction with Matrix Factorization and Universal Schemas
S. Riedel, L. Yao, B. Marlin and A. McCallum. HLT-NAACL, 2013
References

Reasoning With Neural Tensor Networks for Knowledge Base Completion

Modelling Relational Data using Bayesian Clustered Tensor Factorization

Discriminative Probabilistic Models for Relational Data

Knowledge Graph Embedding by Translating on Hyperplanes

Wsabie: Scaling Up To Large Vocabulary Image Annotation

Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction.
J. Weston, A. Bordes, O. Yakhnenko & N. Usunier. *EMNLP*, 2013