COMP 786 (Fall 2020)
Natural Language Processing

Lecture 2 (Aug 19): LMs, Embeddings, Classification

Mohit Bansal

(various slides adapted/borrowed from courses by Dan Klein, Richard Socher, Chris Manning, others)
Language Modeling

- A language model is a distribution over sequences of words (sentences)
  \[ P(w) = P(w_1 \ldots w_n) \]

- Purpose is to usually assign high weights to plausible sentences, e.g., in speech recognition or machine translation

- Also used for language generation now (predict next word given previous words), esp. w/ new RNN models
Traditional N-gram LMs

- Use chain rule to generate words left-to-right

\[ P(w_1 \ldots w_n) = \prod_i P(w_i|w_1 \ldots w_{i-1}) \]

- Can’t condition on the entire left context

\[ P(??? | \text{Turn to page 134 and look at the picture of the}) \]

- N-gram models make a Markov assumption

\[ P(w_1 \ldots w_n) = \prod_i P(w_i|w_{i-k} \ldots w_{i-1}) \]

\[ P(\text{please close the door}) = \]

\[ P(\text{please}|\text{START})P(\text{close}|\text{please}) \ldots P(\text{STOP}|\text{door}) \]
Traditional N-gram LMs

- How do we know \( P(w \mid \text{history}) \)?
  - Use statistics from data (examples using Google N-Grams)
  - E.g. what is \( P(\text{door} \mid \text{the}) \)?

<table>
<thead>
<tr>
<th>Training Counts</th>
<th>198015222 the first</th>
<th>194623024 the same</th>
<th>168504105 the following</th>
<th>158562063 the world</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14112454 the door</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>--------------------</td>
<td>---------------------</td>
<td>--------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>23135851162 the *</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162} = 0.0006
\]

- This is the \textit{maximum likelihood} estimate

https://netspeak.org/#q=*&+the+door&corpus=web-en
Sparsity Issue & Parameter Estimation

- New words all the time (*antidisestablishmentarianism*, *kakorrhaphiophobia,*, www.xyzabc156.com)...worse for new bigrams and trigrams!

  - Maximum likelihood estimates won't get us very far

  \[ \hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')} \]

  - Need to *smooth* these estimates

  - General method (procedurally)
    - Take your empirical counts
    - Modify them in various ways to improve estimates

  - General method (mathematically)
    - Often can give estimators a formal statistical interpretation
    - ... but not always
    - Approaches that are mathematically obvious aren't always what works

---

3516 wipe off the excess
1034 wipe off the dust
547 wipe off the sweat
518 wipe off the mouthpiece
...
120 wipe off the grease
0 wipe off the sauce
0 wipe off the mice
-------------
28048 wipe off the *
We often want to make estimates from sparse statistics:

Smoothing flattens spiky distributions so they generalize better

Very important all over NLP, but easy to do badly!
Smoothing Techniques

- Classic Solution: add-one or add small priors to numer/denom
- Backing off to smaller n-grams
- Held-out Reweighting: Important to optimize/estimate how models generalize! So use held-out data to estimate the map of old count to new count
- Kneser-Ney Discounting: two successful ideas:
  - Idea 1: observed n-grams occur more in training than they will later
  - Idea 2: Type-based fertility (based on how common the word type is)
- Read Chen and Goodman, 1996 for various details and graphs!

http://wordnetweb.princeton.edu/perl/webwn?s=bank
RNN Language Models

- Avoid huge number of n-grams; Memory requirement only scales with #words
- Can condition on all previous history (with forget gates)
- Loss function on identity of predicted word at each time step
- But harder/slower to train and reach optimum (and less interpretable)?
Distributional Semantics

- Words occurring in similar context have similar linguistic behavior (meaning) [Harris, 1954; Firth, 1957]

- Traditional approach: context-counting vectors
  - Count left and right context in window
  - Reweight with PMI or LLR
  - Reduce dimensionality with SVD or NNMF
  
  [Pereira et al., 1993; Lund & Burgess, 1996; Lin, 1998; Lin and Pantel, 2001; Sahlgren, 2006; Pado & Lapata, 2007; Turney and Pantel, 2010; Baroni and Lenci, 2010]

- More word representations: hierarchical clustering based on bigram LM LL
  
  [Brown et al., 1992]
Unsupervised Embeddings

- Vector space representations learned on unlabeled linear context (i.e., left/right words): distributional semantics (Harris, 1954; Firth, 1957)
Newer approach: context-predicting vectors (NNs)

SEenna [Collobert and Weston, 2008; Collobert et al., 2011]: Multi-layer DNN w/ ranking-loss objective; BoW and sentence-level feature layers, followed by std. NN layers. Similar to [Bengio et al., 2003].

\[ \hat{P}(w_t = i \mid \text{context}) \]

\[ \text{softmax} \]

\[ \text{tanh} \]

\[ C(w_{t-n+1}) \]

\[ C(w_{t-2}) \]

\[ C(w_{t-1}) \]

Table look-up in \( C \)

Matrix \( C \)

shared parameters across words

index for \( w_{t-n+1} \)

index for \( w_{t-2} \)

index for \( w_{t-1} \)
Distributional Semantics -- NNs

- CBOW, SKIP, word2vec [Mikolov et al., 2013]: Simple, super-fast NN w/ no hidden layer. Continuous BoW model predicts word given context, skip-gram model predicts surrounding context words given current word.

Other: [Mnih and Hinton, 2007; Turian et al., 2010]

Skipgram word2vec

INPUT  PROJECTION  OUTPUT

$w(t)$  $w(t-2)$  $w(t-1)$

$w(t+1)$  $w(t+2)$

context window $w$

Few mins. vs. days/weeks/months!!

[Mikolov et al., 2013]
Skip-gram word2vec Objective Function

Objective of Skip-gram model is to max. the avg. log probability:

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

The above conditional probability is defined via the softmax function:

$$p(w_O | w_I) = \frac{\exp \left( v'_o \top v_{w_I} \right)}{\sum_{w=1}^{W} \exp \left( v'_w \top v_{w_I} \right)}$$

where $v$ and $v'$ are the “input” and “output” vector representations of $w$, and $W$ is the number of words in the vocabulary.

[mi2013]
Efficient Skip-gram word2vec:

- **Negative Sampling:**

  \[
  \log \sigma(v'_{w_o} \top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i} \top v_{w_I}) \right]
  \]

- I.e., to distinguish the target word \( w_o \) from draws from the noise distribution \( P_n(w) \) using logistic regression, where there are \( k \) negative samples for each data sample.
Efficient Skip-gram word2vec:

Hierarchical Softmax:

\[
p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( \left[ n(w, j + 1) = \text{ch}(n(w, j)) \right] \cdot v'_{n(w, j)}^\top v_{w_I} \right)
\]

Instead of evaluating \( W \) output nodes in the neural network to obtain the probability distribution, it is needed to evaluate only about \( \log_2(W) \) nodes.

Uses a binary tree representation of the output layer with the \( W \) words as its leaves and, for each node, explicitly represents the relative probabilities of its child nodes. These define a random walk that assigns probabilities to words.
Analogy Properties Learned

[Mikolov et al., 2013]
Analogy Properties Learned

[Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.]
Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.
Analogy Properties Learned

Table 4: Examples of the closest entities to the given short phrases, using two different models.

<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.
Distributional Semantics

- Other approaches: spectral methods, e.g., CCA
  - Word-context correlation [Dhillon et al., 2011, 2012]
  - Multilingual correlation [Faruqui and Dyer, 2014; Lu et al., 2015]

- Multi-sense embeddings [Reisinger and Mooney, 2010; Neelakantan et al., 2014]

- Some later ideas: Train task-tailored embeddings to capture specific types of similarity/semantics, e.g.,
  - Dependency context [Bansal et al., 2014, Levy and Goldberg, 2014]
  - Predicate-argument structures [Hashimoto et al., 2014; Madhyastha et al., 2014]
  - Lexicon evidence (PPDB, WordNet, FrameNet) [Xu et al., 2014; Yu and Dredze, 2014; Faruqui et al., 2014; Wieting et al., 2015]
  - Combining advantages of global matrix factorization and local context window methods [GloVe; Pennington et al., 2014]
Multi-sense Embeddings

Different vectors for each sense of a word

Figure 1: Overview of the multi-prototype approach to near-synonym discovery for a single target word independent of context. Occurrences are clustered and cluster centroids are used as prototype vectors. Note the “hurricane” sense of position (cluster 3) is not typically considered appropriate in WSD.

[Reisinger and Mooney, 2010]

Figure 2: Architecture of Multi-Sense Skip-gram (MSSG) model with window size $R_t = 2$ and $K = 3$. Context $c_t$ of word $w_t$ consists of $w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2}$. The sense is predicted by finding the cluster center of the context that is closest to the average of the context vectors.

[Neelakantan et al., 2014]
Syntactically Tailored Embeddings

- Context window size (SKIP)
  - Smaller window → syntactic/functional similarity
  - Larger window → topical similarity

*The morning flight at the JFK airport was delayed*

Similar effect in distributional representations (Lin and Wu, 2009)
Cluster Examples

▸ SKIP, w = 10:

[attendant, takeoff, airport, carry-on, airplane, flown, landings, flew, fly, cabins, ...]

[maternity, childbirth, clinic, physician, doctor, medical, health-care, day-care, ...]

[transactions, equity, investors, capital, financing, stock, fund, purchases, ...]
Cluster Examples

- **SKIP, w = 1**

  - [Mr., Mrs., Ms., Prof., III, Jr., Dr.]
  - [Jeffrey, William, Dan, Robert, Stephen, Peter, John, Richard, ...]
  - [Portugal, Iran, Cuba, Ecuador, Greece, Thailand, Indonesia, ...]

  - [his, your, her, its, their, my, our]
  - [Your, Our, Its, My, His, Their, Her]
  - [truly, wildly, politically, financially, completely, potentially, ...]

[Bansal et al., 2014]
Syntactically Tailored Embeddings

- Syntactic context (\text{SKIP}_{\text{DEP}})
  - Condition on dependency context instead of linear
  - First parse a large corpus with baseline parser:

... said that the \textit{regulation of safety} is ...
Syntactically Tailored Embeddings

- Syntactic context ( SKIP_{DEP} )
  - Condition on dependency context instead of linear
  - Then convert each dependency to a tuple:

\[
\begin{array}{cccccc}
\text{dep label} & \text{grandparent} & \text{parent} & \text{child} & \text{dep label} \\
\text{PMOD}_{<L>} & \text{regulation}_{<G>} & \text{of} & \text{safety} & \text{PMOD}_{<L>} \\
\end{array}
\]

- Syntactic information in clustering, topic, semantic space models
  (Sagae and Gordon, 2009; Haffari et al., 2011; Grave et al., 2013; Boyd-Graber and Blei, 2008; Pado and Lapata, 2007)
# Intrinsic Evaluation

(Finkelstein et al., 2002)

<table>
<thead>
<tr>
<th>Representation</th>
<th>SIM</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BROWN</strong></td>
<td>–</td>
<td><strong>89.3</strong></td>
</tr>
<tr>
<td><strong>SENNA</strong></td>
<td>49.8</td>
<td>85.2</td>
</tr>
<tr>
<td><strong>HUANG</strong></td>
<td><strong>62.6</strong></td>
<td>78.1</td>
</tr>
<tr>
<td><strong>SKIP, w = 10</strong></td>
<td>44.6</td>
<td>71.5</td>
</tr>
<tr>
<td><strong>SKIP, w = 5</strong></td>
<td>44.4</td>
<td>81.1</td>
</tr>
<tr>
<td><strong>SKIP, w = 1</strong></td>
<td>37.8</td>
<td>86.6</td>
</tr>
<tr>
<td><strong>SKIP_{DEP}</strong></td>
<td>34.6</td>
<td>88.3</td>
</tr>
</tbody>
</table>

[1] Bansal et al., 2014
## Parsing Experiments

[Bansal et al., 2014]

### Main WSJ results:

<table>
<thead>
<tr>
<th>System</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>91.9</td>
</tr>
<tr>
<td>BROWN</td>
<td>92.7</td>
</tr>
<tr>
<td>SENNA</td>
<td>92.3</td>
</tr>
<tr>
<td>TURIAN</td>
<td>92.3</td>
</tr>
<tr>
<td>HUANG</td>
<td>92.4</td>
</tr>
<tr>
<td>SKIP</td>
<td>92.3</td>
</tr>
<tr>
<td>SKIP_DEP</td>
<td>92.7</td>
</tr>
</tbody>
</table>

**Ensemble Results**

| ALL – BROWN | 92.9 |
| ALL         | 93.0 |

(faster)

(complementary)
Task-Trained Embeddings

Can also directly train word embeddings on the task, via back-prop from the task supervision (XE errors), e.g., dependency parsing:

\[
y = \text{softmax}(Uh + b_2)
\]

\[
h = \text{ReLU}(Wx + b_1)
\]

Softmax probabilities

Cross-entropy error will be back-propagated to the embeddings.
Multilingual Embeddings via CCA

- Translational context (say, English ↔ German) can help learn stronger embeddings, e.g., separate antonyms vs. synonyms

- CCA on translation pairs to map them to shared space

\[
\begin{align*}
\max_{\mathbf{u} \in \mathbb{R}^{D_x}, \mathbf{v} \in \mathbb{R}^{D_y}} & \quad \frac{\mathbb{E} [(\mathbf{u}^\top \mathbf{x})(\mathbf{v}^\top \mathbf{y})]}{\sqrt{\mathbb{E} [(\mathbf{u}^\top \mathbf{x})^2]} \sqrt{\mathbb{E} [(\mathbf{v}^\top \mathbf{y})^2]}} \\
& = \frac{\mathbf{u}^\top \mathbf{\Sigma}_{xy} \mathbf{v}}{\sqrt{\mathbf{u}^\top \mathbf{\Sigma}_{xx} \mathbf{u}} \sqrt{\mathbf{v}^\top \mathbf{\Sigma}_{yy} \mathbf{v}}}
\end{align*}
\]

[Faruqui and Dyer, 2014]
Multi-view Embeddings via CCA

Table 1: Spearman's correlation (left) and accuracy (right) on different tasks.

<table>
<thead>
<tr>
<th>Language</th>
<th>Before CCA</th>
<th>After CCA</th>
<th>Before CCA</th>
<th>After CCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>En</td>
<td>640</td>
<td>46.7</td>
<td>56.2</td>
<td>36.5</td>
</tr>
<tr>
<td>De-En</td>
<td>512</td>
<td>68.0</td>
<td>74.4</td>
<td>64.6</td>
</tr>
<tr>
<td>Fr-En</td>
<td>512</td>
<td>68.4</td>
<td>73.3</td>
<td>65.7</td>
</tr>
<tr>
<td>Es-En</td>
<td>512</td>
<td>67.2</td>
<td>71.6</td>
<td>64.5</td>
</tr>
</tbody>
</table>

Average: 56.6, 64.5, 51.0, 62.0, 65.5, 60.8, 44.0

5.4 Qualitative Example

To understand how multilingual evidence leads to better results in semantic evaluation tasks, we plot the word representations obtained in §3 of several synonyms and antonyms of the word “beautiful” by projecting both the transformed and untransformed vectors onto $\mathbb{R}^2$ using the t-SNE tool (van der Maaten and Hinton, 2008). The untransformed LSA vectors are in the upper part of Fig. 2, and the CCA-projected vectors are in the lower part. By comparing the two regions, we see that in the untransformed representations, the antonyms are in two clusters separated by the synonyms, whereas in the transformed representation, both the antonyms and synonyms are in their own cluster. Furthermore, the average intra-class distance between synonyms and antonyms is reduced.
Linear vs Deep CCA

Linear CCA results:

<table>
<thead>
<tr>
<th>Embeddings</th>
<th>WS-353</th>
<th>WS-SIM</th>
<th>WS-REL</th>
<th>SL-999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>46.7</td>
<td>56.3</td>
<td>36.6</td>
<td>26.5</td>
</tr>
<tr>
<td>CCA-1</td>
<td>67.2</td>
<td>73.0</td>
<td>63.4</td>
<td>40.7</td>
</tr>
<tr>
<td>CCA-Ens</td>
<td>67.5</td>
<td>73.1</td>
<td>63.7</td>
<td>40.4</td>
</tr>
</tbody>
</table>

- Linear feature mapping not sufficiently powerful to capture hidden, non-linear relationships within data
- Use deep NNs to learn non-linear transformations of orig. embeddings to space where linear correlation maximized
Deep-CCA

2.2 Deep Canonical Correlation Analysis

Both views (Bach and Jordan, 2005).

In a linear CCA setup, projections reconstruct the latent variables that generate multi-dimensional projections by optimizing the view covariance matrices. (1) is extended to learn transformations of each view via deep networks.

Deep-CCA allows a powerful to faithfully capture the hidden, non-linear relationships within the data. Recently, Andrew et al. (2013) proposed a nonlinear extension of CCA between outputs in the two views, measured by a linear CCA step with projection mappings as those of (Faruqui and Dyer, 2014).

These stochastic gradients to update the neural network weights via backpropagation.

KCCA requires computing kernels rather than neural networks. DCCA scales better with data size, as KCCA requires computing kernels rather than neural networks. DCCA scales as stochastic gradient slicing via backpropagation.

The neural network weights and the linear projection mappings are used to extract features from these samples, so as to obtain the two networks, and where

DCCA goals are estimates of the covariance matrices computed for this purpose.

The neural network weights and the linear projection mappings are used to extract features from these samples, so as to obtain the two networks, and where

DCCA goals are estimates of the covariance matrices computed for this purpose.

foul magnificent beastly ugly marvelous elegant
awful horrid grotesque charming splendid

torsors
grotesque magnificent hideous ugly
cute handsome splendid pretty

genzechte aufzuklären schrecklichen elegante
großartige hervorragende wunderbaren

[Lu, Wang, Bansal, Gimpel, Livescu, 2015]
Deep-CCA

- 2 DNNs $f, g$ extract features from the 2 input views $x$ and $y$
- DNNs are trained to maximize output linear correlation of 2 views
- DNN weights and linear projections optimized together:
  \[
  \max_{W_f, W_g, u, v} \frac{u^\top \Sigma_{fg} v}{\sqrt{u^\top \Sigma_{ff} u} \sqrt{v^\top \Sigma_{gg} v}}
  \]
- Covariance matrices computed for $\{f(x_i), g(y_i)\}_{i=1}^N$, as in CCA
- Mini-batch SGD: Feed-forward a sample to estimate $(u, v)$ and gradient and then update NN weights via back-propagation
Results

- Word-similarity improvements

<table>
<thead>
<tr>
<th>Embeddings</th>
<th>WS-353</th>
<th>WS-SIM</th>
<th>WS-REL</th>
<th>SL-999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>46.7</td>
<td>56.3</td>
<td>36.6</td>
<td>26.5</td>
</tr>
<tr>
<td>CCA-1</td>
<td>67.2</td>
<td>73.0</td>
<td>63.4</td>
<td>40.7</td>
</tr>
<tr>
<td>CCA-Ens</td>
<td>67.5</td>
<td>73.1</td>
<td>63.7</td>
<td>40.4</td>
</tr>
<tr>
<td>DCCA-1 (BestAvg)</td>
<td>69.6</td>
<td>73.9</td>
<td>65.6</td>
<td>38.9</td>
</tr>
<tr>
<td>DCCA-Ens (BestAvg)</td>
<td><strong>70.8</strong></td>
<td><strong>75.2</strong></td>
<td><strong>67.3</strong></td>
<td>41.7</td>
</tr>
<tr>
<td>DCCA-1 (MostBeat)</td>
<td>68.6</td>
<td>73.5</td>
<td>65.7</td>
<td>42.3</td>
</tr>
<tr>
<td>DCCA-Ens (MostBeat)</td>
<td>69.9</td>
<td>74.4</td>
<td>66.7</td>
<td>42.3</td>
</tr>
</tbody>
</table>

- Also gets improvements on bigram similarity datasets
Analysis

- High-similarity word pairs that change most with DCCA

<table>
<thead>
<tr>
<th>better with DCCA</th>
<th>worse with DCCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrive come</td>
<td>author creator</td>
</tr>
<tr>
<td>locate find</td>
<td>leader manager</td>
</tr>
<tr>
<td>way manner</td>
<td>buddy companion</td>
</tr>
<tr>
<td>recent new</td>
<td>crowd bunch</td>
</tr>
<tr>
<td>take obtain</td>
<td>achieve succeed</td>
</tr>
<tr>
<td>boundary border</td>
<td>attention interest</td>
</tr>
<tr>
<td>win accomplish</td>
<td>join add</td>
</tr>
<tr>
<td>contemplate think</td>
<td>mood emotion</td>
</tr>
</tbody>
</table>

- DCCA discards hypernymy, separates senses

[Lu, Wang, Bansal, Gimpel, Livescu, 2015]
Analysis

DCCA more cleanly separates synonym-antonym lists

[Lu, Wang, Bansal, Gimpel, Livescu, 2015]
Retrofitting Word Embeddings to Lexicons

- We want the inferred word vector to be close to the observed value \( \hat{q} \) and close to its neighbors \( q_j \), \( \forall j \) such that \((i, j) \in E\), where \( E \) is the set of relations in a dictionary/lexicon (e.g., WordNet, PPDB, etc.)

\[
\Psi(Q) = \sum_{i=1}^{n} \left[ \alpha_i \| q_i - \hat{q}_i \|^2 + \sum_{(i,j) \in E} \beta_{ij} \| q_i - q_j \|^2 \right]
\]

Figure 1: Word graph with edges between related words showing the observed (grey) and the inferred (white) word vector representations.
### Bias in Word Embeddings

<table>
<thead>
<tr>
<th>Extreme <em>she</em></th>
<th>Extreme <em>he</em></th>
<th>Gender stereotype <em>she-he</em> analogies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. homemaker</td>
<td>1. maestro</td>
<td>sewing-carpentry registered nurse-physician housewife-shopkeeper</td>
</tr>
<tr>
<td>2. nurse</td>
<td>2. skipper</td>
<td>nurse-surgeon interior designer-architect softball-baseball</td>
</tr>
<tr>
<td>3. receptionist</td>
<td>3. protege</td>
<td>blond-burly feminism-conservatism cosmetics-pharmaceuticals</td>
</tr>
<tr>
<td>4. librarian</td>
<td>4. philosopher</td>
<td>giggle-chuckle vocalist-guitarist petite-lanky</td>
</tr>
<tr>
<td>5. socialite</td>
<td>5. captain</td>
<td>sassy-snappy diva-superstar charming-affable</td>
</tr>
<tr>
<td>6. hairdresser</td>
<td>6. architect</td>
<td>volleyball-football cupcakes-pizzas lovely-brilliant</td>
</tr>
<tr>
<td>7. nanny</td>
<td>7. financier</td>
<td>gender neutral words while respecting the definitions of these gender specific words.</td>
</tr>
<tr>
<td>8. bookkeeper</td>
<td>8. warrior</td>
<td>gender appropriate <em>she-he</em> analogies</td>
</tr>
<tr>
<td>9. stylist</td>
<td>9. broadcaster</td>
<td>queen-king sister-brother mother-father</td>
</tr>
<tr>
<td>10. housekeeper</td>
<td>10. magician</td>
<td>waitress-waiter ovarian cancer-prostate cancer convent-monastery</td>
</tr>
</tbody>
</table>

Figure 1: **Left** The most extreme occupations as projected on to the *she—he* gender direction on w2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded. **Right** Automatically generated analogies for the pair *she-he* using the procedure described in text. Each automatically generated analogy is evaluated by 10 crowd-workers to whether or not it reflects gender stereotype.

- Debiasing word embeddings via identifying pairs (sets) of words to correct/neutralize, identify bias direction (subspace), and then debias via neutralize+equalize or soften algorithms.

[ Bolukbasi et al., 2016 ]
Compositional Semantics with NNs

- Composing, combining word vectors to representations for longer units: phrases, sentences, paragraphs, ...

- Initial approaches: point-wise sum, multiplication
  [Mitchell and Lapata, 2010; Blacoe and Lapata, 2012]

- Vector-matrix compositionality
  [Baroni and Zamparelli, 2010; Zanzotto et al., 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2011; Yessenalina and Cardie, 2011]

- Linguistic information added via say parses in RvNNs

- Sequential RNNs (with GRU/LSTM gates)
  (Simple vector averaging w/ updating sometimes competitive)
Compositional Semantics with NNs

- Feed-forward NNs with back-propagation

**Softmax (= logistic regression) is not very powerful**

- Softmax only linear decision boundaries

→ Lame when problem is complex

Wouldn’t it be cool to get these correct?

NN and backprop slides from CS224d – Richard Socher
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

Neural Nets for the Win!

- Neural networks can learn much more complex functions and nonlinear decision boundaries!
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

A neuron is essentially a binary logistic regression unit.

\[ h_{w,b}(x) = f(w^T x + b) \]

\[ f(z) = \frac{1}{1 + e^{-z}} \]

- \( w, b \) are the parameters of this neuron, i.e., this logistic regression model.
- \( b \): We can have an “always on” feature, which gives a class prior, or separate it out, as a bias term.
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

A neural network
= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network....
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

Training with Backpropagation

- Let’s consider the derivative of a single weight $W_{ij}$

$$\frac{\partial s}{\partial W} = \frac{\partial}{\partial W} U^T a = \frac{\partial}{\partial W} U^T f(z) = \frac{\partial}{\partial W} U^T f(W x + b)$$

- This only appears inside $a_i$

- For example: $W_{23}$ is only used to compute $a_2$
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

Training with Backpropagation

\[
\frac{\partial s}{\partial W} = \frac{\partial}{\partial W} U^T a = \frac{\partial}{\partial W} U^T f(z) = \frac{\partial}{\partial W} U^T f(Wx + b)
\]

Derivative of weight \( W_{ij} \):

\[
\frac{\partial}{\partial W_{ij}} U^T a \rightarrow \frac{\partial}{\partial W_{ij}} U_i a_i
\]

\[
U_i \frac{\partial}{\partial W_{ij}} a_i = U_i \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}}
\]

\[
= U_i \frac{\partial f(z_i)}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}}
\]

\[
= U_i f'(z_i) \frac{\partial z_i}{\partial W_{ij}}
\]

\[
= U_i f'(z_i) \frac{\partial W_{ij}}{\partial W_{ij}}
\]

\[
= U_i f'(z_i) \frac{\partial W_{ij}}{\partial W_{ij}}
\]

Diagram:

- \( s \) is the output node.
- \( U_2 \) is an input node.
- \( W_{23} \) is the weight connecting \( U_2 \) to \( s \).
- \( a_1, a_2 \) are intermediate nodes.
- \( +1 \) is a bias node.
- \( x_1, x_2, x_3 \) are input nodes.
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

Training with Backpropagation

Derivative of single weight $W_{ij}$:

$$z_i = W_{i}x + b_i = \sum_{j=1}^{3} W_{ij}x_j + b_i$$

$$a_i = f(z_i)$$

\[
U_i \frac{\partial}{\partial W_{ij}} a_i = U_i f'(z_i) \frac{\partial W_{i}x + b_i}{\partial W_{ij}}
\]

\[
= U_i f'(z_i) \frac{\partial}{\partial W_{ij}} \sum_k W_{ik} x_k
\]

\[
= U_i f'(z_i) x_j
\]

\[
= \delta_i x_j
\]

Local error signal

Local input signal

where $f'(z) = f(z)(1 - f(z))$ for logistic $f$
Syntactically Recursive NNs

Socher et al., 2013a, 2014: RvNNs on constituent and dependency parse trees

https://stanfordnlp.github.io/CoreNLP/demo.html
https://parser.kitaev.io/
Recurrent NNs

- Recurrent NNs (RNNs) are non-tree, sequential versions of recursive RvNNs
- Weights tied together for each time step
- Loss function on identity of predicted word at each time step
LSTM RNNs

- LSTM (Long short term memory) RNNs have gates for forgetting, allowing learning of longer-term connections by avoiding vanishing/exploding gradients

https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Character RNNs

- Can directly process each character as a unit!
- Helps learn prefixes, stems, suffixes (form vs. function, rare/unseen words, etc.)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Supervised Sentence Embedding Models

Just like word embeddings were supervised using lexicons, dictionaries, taxonomies (WordNet) etc., sentence embeddings also benefit greatly from supervision!

2 examples: supervision based on bidirectional sentence similarity (paraphrases) or directed similarity (entailment vs contradiction vs neutral)
Paraphrase-based Sentence Embeddings

Phrases that mean the same, are replaceable in context

- main reason why
- informed about the outcome
- with particular emphasis
- we 'll have a good time
- 50 years ago
- that , according to
- program is aimed at
- are under the obligation
- a critical component
- principal reason for
- notified of the results
- with specific focus
- we 're gonna have fun
- five decades ago
- which , in accordance with
- programme aims to
- have a duty
- an essential element
Paraphrase-based Sentence Embeddings

- PPDB: Massive, useful resource (220M) automatically extracted from parallel bilingual corpora [Ganitkevitch et al., 2013]

- Idea summary: carefully extract a few (< 0.05%) +ve and -ve pairs from unannotated PPDB as weak supervision

- Train a parametric paraphrase model (2-view RNN with hinge loss) on these pairs, to be able to represent arbitrary phrases as embeddings

- This learns strong word/phrase embeddings that better predict paraphrases on new annotated PPDB subset and gets SoA on word/bigram similarity datasets

[Wieting, Bansal, Gimpel, Livescu, Roth, 2015]
2 parse-based RvNNNs with a hinge-based loss function

Composition =
\[ g(p) = f(W[g(c_1); g(c_2)] + b) \]
Paraphrase Model

- Loss: +ve pairs closer than -ve pairs with margin $\delta$

Positive training pairs

$$\min_{W,b,W_w} \frac{1}{|X|} \left( \sum_{\langle x_1,x_2 \rangle \in X} \max(0, \delta - g(x_1) \cdot g(x_2)) + \max(0, \delta - g(x_1) \cdot g(t_1)) \right)$$

Negative training pairs

$$+ \max(0, \delta - g(x_1) \cdot g(x_2)) + \max(0, \delta - g(x_2) \cdot g(t_2))$$

Regularization terms

[Wieting, Bansal, Gimpel, Livescu, Roth, 2015]
Entailment-based Embeddings

- SNLI and Multi-NLI corpora with sentence pairs of 3 relationships: entailment, contradiction, neutral/unrelated

<table>
<thead>
<tr>
<th>Premise</th>
<th>Label</th>
<th>Hypothesis</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Old One always comforted Ca'daan, except today.</td>
<td>neutral</td>
<td>Ca'daan knew the Old One very well.</td>
<td>Fiction</td>
</tr>
<tr>
<td>Your gift is appreciated by each and every student who will benefit from your generosity.</td>
<td>neutral</td>
<td>Hundreds of students will benefit from your generosity.</td>
<td>Letters</td>
</tr>
<tr>
<td>yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or</td>
<td>contradiction</td>
<td>August is a black out month for vacations in the company.</td>
<td>Telephone Speech</td>
</tr>
<tr>
<td>At the other end of Pennsylvania Avenue, people began to line up for a White House tour.</td>
<td>entailment</td>
<td>People formed a line at the end of Pennsylvania Avenue.</td>
<td>9/11 Report</td>
</tr>
<tr>
<td>A black race car starts up in front of a crowd of people.</td>
<td>contradiction</td>
<td>A man is driving down a lonely road.</td>
<td>SNLI</td>
</tr>
</tbody>
</table>

[Bowman et al., 2015; Williams et al., 2017]
Entailment-based Embeddings

Encoding

Premise  →  Encoder  →  $v$

Hypothesis  →  Encoder  →  $u$

Matching

Same Structure

$[v, u, v \odot u, |v - u|]$

[Conneau et al., 2017]
### Entailment-based Embeddings

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>CR</th>
<th>SUBJ</th>
<th>MPQA</th>
<th>SST</th>
<th>TREC</th>
<th>MRPC</th>
<th>SICK-R</th>
<th>SICK-E</th>
<th>STS14</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsupervised representation training (unordered sentences)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram-TFIDF</td>
<td>73.7</td>
<td><strong>92.9</strong></td>
<td>90.3</td>
<td>82.4</td>
<td>-</td>
<td>85.0</td>
<td>73.6/81.7</td>
<td>-</td>
<td>-</td>
<td>.58/.57</td>
</tr>
<tr>
<td>ParagraphVec (DBOW)</td>
<td>60.2</td>
<td>66.9</td>
<td>76.3</td>
<td>70.7</td>
<td>-</td>
<td>59.4</td>
<td>72.9/81.1</td>
<td>-</td>
<td>-</td>
<td>.42/.43</td>
</tr>
<tr>
<td>SDAE</td>
<td>74.6</td>
<td>78.0</td>
<td>90.8</td>
<td>86.9</td>
<td>-</td>
<td>78.4</td>
<td><strong>73.7</strong>/80.7</td>
<td>-</td>
<td>-</td>
<td>.37/.38</td>
</tr>
</tbody>
</table>
| SIF (GloVe + WR) | - | - | - | - | 82.2 | - | - | - | **84.6** | .69/-
| word2vec BOW† | 77.7 | 79.8 | 90.9 | 88.3 | 79.7 | 83.6 | 72.5/81.4 | 0.803 | 78.7 | .65/.64 |
| fastText BOW† | 76.5 | 78.9 | 91.6 | 87.4 | 78.8 | 81.8 | 72.4/81.2 | 0.800 | 77.9 | .63/62 |
| GloVe BOW† | **78.7** | 78.5 | **91.6** | 87.6 | 79.8 | 83.6 | 72.1/80.9 | 0.800 | 78.6 | .54/56 |
| GloVe Positional Encoding† | 78.3 | 77.4 | 91.1 | 87.1 | 80.6 | 83.3 | 72.5/81.2 | 0.799 | 77.9 | .51/54 |
| BiLSTM-Max (untrained)† | 77.5 | **81.3** | 89.6 | **88.7** | 80.7 | **85.8** | 73.2/81.6 | **0.860** | 83.4 | .39/.48 |
| **Unsupervised representation training (ordered sentences)** | | | | | | | | | | |
| FastSent | 70.8 | 78.4 | 88.7 | 80.6 | - | 76.8 | 72.2/80.3 | - | - | .63/64 |
| FastSent+AE | 71.8 | 76.7 | 88.8 | 81.5 | - | 80.4 | 71.2/79.1 | - | - | .62/62 |
| SkipThought | 76.5 | 80.1 | 93.6 | 87.1 | 82.0 | **92.2** | 73.0/82.0 | **0.858** | 82.3 | .29/35 |
| SkipThought-LN | **79.4** | **83.1** | **93.7** | 89.3 | 82.9 | 88.4 | - | **0.858** | 79.5 | .44/45 |
| **Supervised representation training** | | | | | | | | | | |
| CaptionRep (bow) | 61.9 | 69.3 | 77.4 | 70.8 | - | 72.2 | - | - | - | .46/42 |
| DictRep (bow) | 76.7 | 78.7 | 90.7 | 87.2 | - | 81.0 | 68.4/76.8 | - | - | **.67/.70** |
| NMT En-to-Fr | 64.7 | 70.1 | 84.9 | 81.5 | - | 82.8 | - | - | - | .43/42 |
| Paragram-phrase | - | - | - | - | 79.7 | - | - | 0.849 | 83.1 | **.71/-** |
| BiLSTM-Max (on SST)† | (*) | 83.7 | 90.2 | 89.5 | (*) | 86.0 | 72.7/80.9 | 0.863 | 83.1 | .55/54 |
| BiLSTM-Max (on SNLI)† | 79.9 | 84.6 | 92.1 | **89.8** | 83.3 | **88.7** | 75.1/82.3 | **0.885** | **86.3** | .68/65 |
| BiLSTM-Max (on AllNLI)† | **81.1** | **86.3** | **92.4** | **90.2** | **84.6** | 88.2 | **76.2**/83.1 | **0.884** | **86.3** | **.70/.67** |
| **Supervised methods (directly trained for each task – no transfer)** | | | | | | | | | | |
| Naive Bayes - SVM | 79.4 | 81.8 | 93.2 | 86.3 | 83.1 | - | - | - | - | - |
| AdaSent | 83.1 | 86.3 | 95.5 | 93.3 | - | 92.4 | - | - | - | - |
| TF-KLD | - | - | - | - | - | - | 80.4/85.9 | - | - | - |
| Illinois-LH | - | - | - | - | - | - | - | 84.5 | - | - |
| Dependency Tree-LSTM | - | - | - | - | - | - | 0.868 | - | - | - |

*Note: The table represents the results of various models trained on different datasets and tasks. The models are grouped by nature of data and architectures trained in different ways. The table includes unsupervised and supervised representation training methods, along with their performance metrics for different tasks.*

![Source: Conneau et al., 2017](https://example.com/conneau2017)
Classification Tasks: Sentiment Analysis

Sentiment Analysis with Python NLTK Text Classification

This is a demonstration of sentiment analysis using a NLTK 2.0.4 powered text classification process. It can tell you whether it thinks the text you enter below expresses positive sentiment, negative sentiment, or if it's neutral. Using hierarchical classification, neutrality is determined first, and sentiment polarity is determined second, but only if the text is not neutral.

Analyze Sentiment

Language
english

Enter text

It always amazes me how Universal never cares to create anything remotely clever when it comes to their animations, and so once again they come up with a harmless little story that wants to be cute and funny (which it is sometimes) but is only bound to be quickly forgotten.

Enter up to 50000 characters

Analyze

Sentiment Analysis Results

The text is neg.

The final sentiment is determined by looking at the classification probabilities below.

Subjectivity
- neutral: 0.3
- polar: 0.7

Polarity
- pos: 0.2
- neg: 0.8
Sentiment Analysis

- Earlier methods used bag of words, e.g., lexicons of positive and negative words and phrases

- Cannot distinguish tricky cases like:

  + white blood cells destroying an infection
  - an infection destroying white blood cells

  + There are slow and repetitive parts but it has just enough spice to keep it interesting.
  - Stealing Harvard doesn’t care about cleverness, wit or any other kind of intelligent humor.
Sentiment Analysis

- Even simpler issues like negation hard to understand
- Socher et al., 2013b present new compositional training data and new composition model

![Recursive Neural Tensor Network diagram](image-url)
Sentiment Analysis

- Even simpler issues like negation hard to understand
- Socher et al., 2013b present new compositional training data and new composition model
Sentiment Analysis

Sentiment Compositionality:

- Parse trees of 11,855 sentences
- 215,154 phrases with labels
- Allows training and evaluating with compositional information
Sentiment Analysis

Better Models: Tree-based LSTM-RNNs

Tree LSTMs

- We can use those ideas in grammatical tree structures!
- Paper: Tai et al. 2015:
  Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks
- Idea: Sum the child vectors in a tree structure
- Each child has its own forget gate
- Same softmax on $h$

$$\tilde{h}_j = \sum_{k \in C(j)} h_k,$$
$$i_j = \sigma \left( W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right),$$
$$f_{jk} = \sigma \left( W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right),$$
$$o_j = \sigma \left( W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right),$$
$$u_j = \tanh \left( W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right),$$
$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k,$$
$$h_j = o_j \odot \tanh(c_j),$$
Sentiment Analysis

- Better Models: Recursive Neural Tensor Network (RNTN)

Socher et al. 2013
Sentiment Compositionality

Results on Stanford Sentiment Treebank

<table>
<thead>
<tr>
<th>Method</th>
<th>Fine-grained</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAE (Socher et al., 2013)</td>
<td>43.2</td>
<td>82.4</td>
</tr>
<tr>
<td>MV-RNN (Socher et al., 2013)</td>
<td>44.4</td>
<td>82.9</td>
</tr>
<tr>
<td>RNTN (Socher et al., 2013)</td>
<td>45.7</td>
<td>85.4</td>
</tr>
<tr>
<td>DCNN (Blunsom et al., 2014)</td>
<td>48.5</td>
<td>86.8</td>
</tr>
<tr>
<td>Paragraph-Vec (Le and Mikolov, 2014)</td>
<td>48.7</td>
<td>87.8</td>
</tr>
<tr>
<td>CNN-non-static (Kim, 2014)</td>
<td>48.0</td>
<td>87.2</td>
</tr>
<tr>
<td>CNN-multichannel (Kim, 2014)</td>
<td>47.4</td>
<td>88.1</td>
</tr>
<tr>
<td>DRNN (Irsoy and Cardie, 2014)</td>
<td>49.8</td>
<td>86.6</td>
</tr>
<tr>
<td>LSTM</td>
<td>45.8</td>
<td>86.7</td>
</tr>
<tr>
<td>Bidirectional LSTM</td>
<td>49.1</td>
<td>86.8</td>
</tr>
<tr>
<td>2-layer LSTM</td>
<td>47.5</td>
<td>85.5</td>
</tr>
<tr>
<td>2-layer Bidirectional LSTM</td>
<td>46.2</td>
<td>84.8</td>
</tr>
<tr>
<td>Constituency Tree LSTM (no tuning)</td>
<td>46.7</td>
<td>86.6</td>
</tr>
<tr>
<td>Constituency Tree LSTM</td>
<td><strong>50.6</strong></td>
<td>86.9</td>
</tr>
</tbody>
</table>


[Yessenalina and Cardie, 2011; Socher et al., 2013b]
Other Classification Tasks

- Sentence similarity
- Entailment classification
- Spam detection
- Document topic classification
- Others: abusive language, hate speech, humor, irony, rumor, sarcasm detection, etc.

SemEval has great new tasks every year with novel datasets in many cases! Some recent years: