COMP 790.139 (Fall 2017)
Natural Language Processing
(with deep learning and connections to vision/robotics)

Mohit Bansal

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

- Chapter section summaries were due yesterday
- Make sure you regularly check your ConnectCarolina email id’s
- 1\textsuperscript{st} Coding assignment to be out soon
- Start thinking of projects early!

- TA: Yixin Nie (yixin1@cs.unc.edu) -- will announce office hours soon!
Recap of 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

- 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)

**SENNA** [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].

\[
i\text{-th output} = P(w_t = i \mid \text{context})
\]

\[
C(w_{t-n+1}) \quad \cdots \quad C(w_{t-2}) \quad C(w_{t-1})
\]

Table look-up in \( C \)

Matrix \( C \)

\[
\text{shared parameters across words}
\]

\[
\text{index for } w_{t-n+1} \quad \text{index for } w_{t-2} \quad \text{index for } w_{t-1}
\]
**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.

  ![Diagram of CBOW and Skip-gram models]

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

Skipgram word2vec

[Mikolov et al., 2013]

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

\[
p(w_O | w_I) = \frac{\exp \left( v'_{wo} \top v_{wI} \right)}{\sum_{w=1}^{W} \exp \left( v'_{w} \top v_{wI} \right)}
\]

where \(v\) and \(v'\) are the “input” and “output” vector representations of \(w\), and \(W\) is the number of words in the vocabulary.
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:**

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

  - 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

Figure 2: Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In high-dimensional space, multiple relations can be embedded for a single word.

We have explored several related methods and found that the proposed method performs well for both syntactic and semantic relations. We note that this measure is qualitatively similar to relational similarity model of (Turney, 2012), which predicts similarity between members of the word pairs \((x_b, x_d), (x_c, x_d)\) and dis-similarity for \((x_a, x_d)\).

### Experimental Results

To evaluate the vector offset method, we used vectors generated by the RNN toolkit of Mikolov (2012). Vectors of dimensionality 80, 320, and 640 were generated, along with a composite of several systems, with total dimensionality 1600. The systems were trained with 320M words of Broadcast News data as described in (Mikolov et al., 2011a), and had an 82k vocabulary.

Table 2 shows results for both RNNLM and LSA vectors on the syntactic task. LSA was trained on the same data as the RNN. We see that the RNN vectors capture significantly more syntactic regularity than the LSA vectors, and do remarkably well in an absolute sense, answering more than one in three questions correctly.

Table 3 compares the RNN vectors with those based on the methods of Collobert and Weston (2008) and Mnih and Hinton (2009), as implemented by (Turian et al., 2010) and available online.

Since different words are present in these datasets, we computed the intersection of the vocabularies of the RNN vectors and the new vectors, and restricted the test set and word vectors to those. This resulted in a 36k word vocabulary, and a test set with 6632 questions. Turian's Collobert and Weston based vectors do poorly on this task, whereas the Hierarchical Log-Bilinear Model vectors of (Mnih and Hinton, 2009) do essentially as well as the RNN vectors. These representations were trained on 37M words of data and this may indicate a greater robustness of the HLBL method.

We conducted similar experiments with the semantic test set. For each target word pair in a relation category, the model measures its relational similarity to each of the prototypical word pairs, and then uses the average as the final score. The results are evaluated using the two standard metrics defined in the task, Spearman's rank correlation coefficient and MaxDiff accuracy. In both cases, larger values are better. To compare to previous systems, we report the average over all 69 relations in the test set.

From Table 4, we see that as with the syntactic regularity study, the RNN-based representations perform best. In this case, however, Turian's CW vectors are comparable in performance to the HLBL vectors. With the RNN vectors, the performance improves as the number of dimensions increases. Surprisingly, we found that even though the RNN vectors capture significantly more syntactic regularity than the LSA vectors, and do remarkably well in an absolute sense, answering more than one in three questions correctly.
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.
Analogy Properties Learned

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 $\rightarrow$ syntactic/functional similarity
  - Larger window $\rightarrow$ 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, ...]

[Bansal et al., 2014]
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, ...]
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 regulation of safety is ...

(grandparent) (parent) (child)
Syntactically Tailored Embeddings

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

\[
\begin{array}{cccccc}
\text{dep label} & \text{grandparent} & \text{parent} & \text{child} & \text{dep label} \\
\end{array}
\]

\[
\begin{array}{cccccc}
PMOD_{<L>} & \text{regulation}_{<G>} & \text{of} & \text{safety} & PMOD_{<L>} \\
\end{array}
\]

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

System Test Baseline

- **BROWN**: 92.7
- **SENN** (Buckets): 92.3
- **HUANG** (Buckets): 92.4
- **SENN** (Hier. Clustering): 92.3
- **HUANG** (Hier. Clustering): 92.4
- **BROWN**: 92.7
- **SENN**: 92.3
- **TURIA** (Hier. Clustering): 92.3
- **HUANG**: 92.4
- **SENN**: 92.3
- **SKIP DEP**: 92.7

Ensemble Results

- **ALL–BROWN**: 84.7
- **ALL**: 84.9

System Test Avg (5 domains)

- **Baseline**: 83.5
- **BROWN**: 84.2
- **SENN**: 84.3
- **TURIA**: 83.9
- **HUANG**: 84.1
- **SENN**: 83.7
- **SKIP DEP**: 84.1

[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><strong>BROWN</strong></td>
<td><strong>92.7</strong></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><strong>SKIP_DEP</strong></td>
<td><strong>92.7</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ensemble Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALL – BROWN</strong></td>
</tr>
<tr>
<td><strong>ALL</strong></td>
</tr>
</tbody>
</table>

(faster)

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

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

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

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

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

- CCA on translation pairs to map them to shared space

\[
\max_{u \in \mathbb{R}^{D_x}, v \in \mathbb{R}^{D_y}} \frac{\mathbb{E} [(u^\top x)(v^\top y)]}{\sqrt{\mathbb{E} [(u^\top x)^2]} \sqrt{\mathbb{E} [(v^\top y)^2]}} = \frac{u^\top \Sigma_{xy} v}{\sqrt{u^\top \Sigma_{xx} u} \sqrt{v^\top \Sigma_{yy} v}}
\]

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

Before CCA

After CCA

[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 $R^2$ 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.

[5.5 Variation in Vector Length]
In order to demonstrate that the gains in performance by using multilingual correlation sustains before CCA

After CCA

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

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>Before CCA</th>
<th>After CCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-En</td>
<td></td>
<td></td>
</tr>
<tr>
<td>De-En</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fr-En</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Es-En</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Figure 2: Monolingual (top) and multilingual (bottom; marked with apostrophe) word projections of the antonyms (shown in red) and synonyms of "beautiful".]

[Figure 3: Performance of monolingual and multilingual vectors on WS-353 for different vector lengths.]

Before CCA

After CCA

[Faruqui and Dyer, 2014]
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
2.2 Deep Canonical Correlation Analysis

Deep-CCA is a method for learning cross-view representations by optimizing the correlations between multi-dimensional projections of different views. It is an extension of Canonical Correlation Analysis (CCA) that can capture non-linear relationships in the data. Given sample pairs \( (x_1, x_2) \), Deep-CCA learns cross-view transformations \( u, v \) of each view via deep networks.

The objective is to find two vectors \( u \) and \( v \) such that the cross-view covariance of \( x_1 u \) and \( x_2 v \) is maximized, while maintaining the within-view covariances of \( x_1 u \) and \( x_2 v \) as small as possible. This is achieved by optimizing the following objective function:

\[
\max_{u, v} \frac{u^\top x_1 \Sigma_{xx} u \cdot v^\top x_2 \Sigma_{yy} v}{\sqrt{(u^\top x_1 \Sigma_{xx} u)^2 + (v^\top x_2 \Sigma_{yy} v)^2}}
\]

where \( \Sigma_{xx} \) and \( \Sigma_{yy} \) are the covariance matrices of \( x_1 \) and \( x_2 \) respectively. The optimization is performed using gradient-based methods, such as stochastic gradient descent.

Deep-CCA differs from linear CCA in that it can learn more complex, non-linear relationships between the views. This is achieved by using deep neural networks to transform the data before applying the CCA step with projection mappings computed for \( x_1 \) and \( x_2 \). The output weights \( \{u, v\} \) are learned by backpropagation, allowing the model to capture the hidden, non-linear relationships in the data.
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

[Andrew et al., 2013]
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><strong>42.3</strong></td>
</tr>
<tr>
<td>DCCA-Ens (MostBeat)</td>
<td>69.9</td>
<td>74.4</td>
<td>66.7</td>
<td><strong>42.3</strong></td>
</tr>
</tbody>
</table>

- Also gets improvements on bigram similarity datasets

[Lu, Wang, Bansal, Gimpel, Livescu, 2015]
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</td>
<td>author</td>
</tr>
<tr>
<td>locate</td>
<td>leader</td>
</tr>
<tr>
<td>way</td>
<td>buddy</td>
</tr>
<tr>
<td>recent</td>
<td>crowd</td>
</tr>
<tr>
<td>take</td>
<td>achieve</td>
</tr>
<tr>
<td>boundary</td>
<td>attention</td>
</tr>
<tr>
<td>win</td>
<td>join</td>
</tr>
<tr>
<td>contemplate</td>
<td>mood</td>
</tr>
<tr>
<td>think</td>
<td>emotion</td>
</tr>
</tbody>
</table>

DCCA discards hypernymy, separates senses

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

- DCCA more cleanly separates synonym-antonym lists

\[
d(d) - (c(w)) + o(w) \leq 0, \text{ if } \Delta(w) < 0, \text{ then the word pair was closer to the human ranking using DCCA.}
\]

Table 3 shows word pairs from SimLex-999 with high human similarity ratings (≥ 7 out of 10); column 1 shows pairs with smallest \(\Delta\) values, and column 2 shows pairs with largest \(\Delta\) values.

Among pairs in column 1, many contain words with several senses. Using bilingual information is likely to focus on the most frequent sense in the bi-text, due to our use of the most frequently-aligned German word in each training pair. By contrast, using only monolingual context is expected to find an embedding that blends the contextual information across all word senses.

Several pairs from column 2 show hypernym rather than paraphrase relationships, e.g., author-creator and leader-manager. Though these pairs are rated as highly similar by annotators, linear CCA made them less similar than the original vectors, and DCCA made them less similar still. This matches our intuition that bilingual information should encourage paraphrase-like similarity and thereby discourage the similarity of hypernym-hyponym pairs.

Visualizations

We visualized several synonym-antonym word lists and often found that DCCA more cleanly separated synonyms from antonyms than CCA or the original vectors. An example of the clearest improvement is shown in Fig. 2.

Related work

Previous work has successfully used translational context for word representations (Diab and Resnik, 2002; Zhao et al., 2005; Täckström et al., 2012; Bansal et al., 2012; Faruqui and Dyer, 2014), including via hand-designed vector space models (Peirsman and Padó, 2010; Sumita, 2000) or via unsupervised LDA and LSA (Boyd-Graber and Blei, 2009; Zhao and Xing, 2006).

There have been other recent deep learning approaches to bilingual representations, e.g., based on a join monolingual and bilingual objective (Zou et al., 2013). There has also been recent interest in learning bilingual representations without using word alignments (Chandar et al., 2014; Gouws et al., 2014; Kocisky et al., 2014; Vulic and Moens, 2013).

This research is also related to early examples of learning bilingual lexicons using monolingual corpora (Koehn and Knight, 2002; Haghighi et al., 2008); the latter used CCA to find matched word pairs. Irvine and Callison-Burch (2013) used a supervised learning method with multiple monolingual signals. Finally, other work on CCA and spectral methods has been used in the context of other types of views (Collobert and Weston, 2008; Dhillon et al., 2011; Klementiev et al., 2012; Chang et al., 2013).

Conclusion

We have demonstrated how bilingual information can be incorporated into word embeddings via deep canonical correlation analysis (DCCA). The DCCA embeddings consistently outperform linear CCA embeddings on word and bigram similarity tasks.

Future work could compare DCCA to other non-linear approaches discussed in §5, compare different languages as multiview context, and extend to aligned phrase pairs, and to unaligned data.

Acknowledgments

We are grateful to Manaal Faruqui for sharing resources, and to Chris Dyer, David Sontag, Lyle Ungar, and anonymous reviewers for helpful input.

[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.

[Faruqui et al., 2015]
Bias in Word Embeddings

<table>
<thead>
<tr>
<th>Extreme she</th>
<th>Extreme he</th>
<th>Gender stereotype she-he 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></td>
</tr>
<tr>
<td>8. bookkeeper</td>
<td>8. warrior</td>
<td></td>
</tr>
<tr>
<td>9. stylist</td>
<td>9. broadcaster</td>
<td></td>
</tr>
<tr>
<td>10. housekeeper</td>
<td>10. magician</td>
<td></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

Wouldn’t it be cool to get these correct?

→ Lame when problem is complex
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!
A neuron is essentially a binary logistic regression unit

\[ h_{w,b}(x) = f(w^T x + b) \]
\[ f(z) = \frac{1}{1 + e^{-z}} \]

*\( b \): We can have an “always on” feature, which gives a class prior, or separate it out, as a bias term

*\( w, b \) are the parameters of this neuron i.e., this logistic regression model
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....
Feed-forward NNs with back-propagation

Training with Backpropagation

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

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

• 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 \( W_{ij} \):

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

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

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

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

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

\[
\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \frac{\partial u}{\partial x}
\]

\[
z_i = W_i x + b_i = \sum_{j=1}^{3} W_{ij} x_j + b_i
\]

\[
a_i = f(z_i)
\]
Compositional Semantics with NNs

Feed-forward NNs with back-propagation

Training with Backpropagation

Derivative of single weight $W_{ij}$:

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

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

\[
= \underbrace{U_i f'(z_i)}_{\delta_i} x_j
\]

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

- Local error signal
- Local input signal

Diagram: feed-forward network with back-propagation.
Syntactically Recursive NNs

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

\[
\begin{align*}
\left(p^{(2)}, p^{(2)}\right) & = f \left(W^{(A, p^{(1)}} \left(a, p^{(1)}\right)\right) \\
\left(p^{(1)}, p^{(1)}\right) & = f \left(W^{(B, c)} \left(b, c\right)\right)
\end{align*}
\]

(A, a=) (B, b=) (C, c=)

Students

Ride

Bikes

At

Night
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.
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

<table>
<thead>
<tr>
<th>Original Phrase</th>
<th>Paraphrase</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>main reason why</td>
<td>principal reason for</td>
<td></td>
</tr>
<tr>
<td>informed about the outcome</td>
<td>notified of the results</td>
<td></td>
</tr>
<tr>
<td>with particular emphasis</td>
<td>with specific focus</td>
<td></td>
</tr>
<tr>
<td>we 'll have a good time</td>
<td>we 're gonna have fun</td>
<td></td>
</tr>
<tr>
<td>50 years ago</td>
<td>five decades ago</td>
<td></td>
</tr>
<tr>
<td>that , according to</td>
<td>which , in accordance with</td>
<td></td>
</tr>
<tr>
<td>program is aimed at</td>
<td>programme aims to</td>
<td></td>
</tr>
<tr>
<td>are under the obligation</td>
<td>have a duty</td>
<td></td>
</tr>
<tr>
<td>a critical component</td>
<td>an essential element</td>
<td></td>
</tr>
</tbody>
</table>
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]
Paraphrase Model

- 2 parse-based RvNNNs with a hinge-based loss function

![Diagram of paraphrase model with parse trees and loss calculation formula]

Composition = $g(p) = f(W[g(c_1); g(c_2)] + b)$

[Socher et al., 2011]
Paraphrase Model

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

\[
\min_{W, b, W_w} \frac{1}{|X|} \left( \sum_{\langle x_1, x_2 \rangle \in X} \max(0, \delta - g(x_1) \cdot g(x_2)) + g(x_1) \cdot g(t_1) \right) + \max(0, \delta - g(x_1) \cdot g(x_2)) + g(x_2) \cdot g(t_2) \right) + \sum_{w \in W} \lambda_w \left( W_w^2 + b_w^2 \right) + \sum_{k=1}^{K} \left( W_k^T W_k + b_k^T b_k \right) + \lambda_{w_{initial}} W_{w_{initial}}^2
\]

[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

Same Structure

Matching

[v, u, v ⊗ u, |v − u|] → MLP → Prediction

[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>79.2</td>
<td>90.3</td>
<td>82.4</td>
<td>-</td>
<td>-</td>
<td>85.0</td>
<td>73.6/81.7</td>
<td>-</td>
<td>-</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>-</td>
<td>59.4</td>
<td>72.9/81.1</td>
<td>-</td>
<td>-</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>-</td>
<td>78.4</td>
<td>73.7/80.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SIF (GloVe + WR)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>82.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>84.6</td>
</tr>
<tr>
<td>word2vec BOW†</td>
<td>77.7</td>
<td>79.8</td>
<td>90.9</td>
<td>88.3</td>
<td>79.7</td>
<td>83.6</td>
<td>72.5</td>
<td>81.4/0.803</td>
<td>78.7</td>
<td>.65/.64</td>
</tr>
<tr>
<td>fastText BOW‡</td>
<td>76.5</td>
<td>78.9</td>
<td>91.6</td>
<td>87.4</td>
<td>78.8</td>
<td>81.8</td>
<td>72.4</td>
<td>81.2/0.800</td>
<td>77.9</td>
<td>.63/62</td>
</tr>
<tr>
<td>GloVe BOW†</td>
<td><strong>78.7</strong></td>
<td>78.5</td>
<td><strong>91.6</strong></td>
<td>87.6</td>
<td>79.8</td>
<td>83.6</td>
<td>72.1</td>
<td>80.9/0.800</td>
<td>78.6</td>
<td>.54/56</td>
</tr>
<tr>
<td>GloVe Positional Encoding‡</td>
<td>78.3</td>
<td>77.4</td>
<td>91.1</td>
<td>87.1</td>
<td>80.6</td>
<td>83.3</td>
<td>72.5</td>
<td>81.2/0.799</td>
<td>77.9</td>
<td>.51/54</td>
</tr>
<tr>
<td>BiLSTM-Max (untrained)‡</td>
<td>77.5</td>
<td><strong>81.3</strong></td>
<td>89.6</td>
<td><strong>88.7</strong></td>
<td>80.7</td>
<td><strong>85.8</strong></td>
<td>73.2</td>
<td>81.6/0.860</td>
<td>83.4</td>
<td>.39/48</td>
</tr>
</tbody>
</table>

| **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  |
| Paragraph-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 | 85.9/0.71  | -      | -      |
| Illinois-LH              | -    | -    | -     | -    | -    | -    | -    | 84.5    | -      | -      |
| Dependency Tree-LSTM     | -    | -    | -     | -    | -    | -    | -    | 0.868   | -      | -      |

---

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

- Encoder details: Starting point is 1-layer biLSTM with Max-pooling
Entailment-based Embeddings

- Improved Encoders: e.g., via shortcut-stacked RNNs (to help learn higher-level semantic features and to help sparse gradients from max-pooling to flow to lower layers

![Diagram of biLSTM model with word embedding, source sentence, and fine-tuning connections.](image-url)

Entailment-based Embeddings

- Improved Encoders: e.g., via shortcut-stacked RNNs (to help learn higher-level semantic features and to help sparse gradients from max-pooling to flow to lower layers

![Diagram of biLSTM model with word embedding, source sentence, and fine-tuning connections.](image-url)
### Entailment-based Embeddings

#### RepEval 2017 Shared Task Results

<table>
<thead>
<tr>
<th>Team Name</th>
<th>Authors</th>
<th>Matched</th>
<th>Mismatched</th>
<th>Model Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha (ensemble)</td>
<td>Chen et al.</td>
<td>74.9%</td>
<td>74.9%</td>
<td>STACK, CHAR, Attn., Pool, ProdDiff</td>
</tr>
<tr>
<td>YixinNie-UNC-NLP</td>
<td>Nie and Bansal</td>
<td>74.5%</td>
<td>73.5%</td>
<td>STACK, Pool, ProdDiff, SNLI</td>
</tr>
<tr>
<td>alpha</td>
<td>Chen et al.</td>
<td>73.5%</td>
<td>73.6%</td>
<td>Attn, Pool, ProdDiff, SNLI</td>
</tr>
<tr>
<td>Rivercorners (ensemble)</td>
<td>Balazs et al.</td>
<td>72.2%</td>
<td>72.8%</td>
<td>Attn, Pool, ProdDiff, SNLI</td>
</tr>
<tr>
<td>Rivercorners</td>
<td>Balazs et al.</td>
<td>72.1%</td>
<td>72.1%</td>
<td>Attn, Pool, ProdDiff, SNLI</td>
</tr>
<tr>
<td>LCT-MALTA</td>
<td>Vu et al.</td>
<td>70.7%</td>
<td>70.8%</td>
<td>Char, EnhEmb, ProdDiff, Pool</td>
</tr>
<tr>
<td>TALP-UPC</td>
<td>Yang et al.</td>
<td>67.9%</td>
<td>68.2%</td>
<td>Char, Attn, SNLI</td>
</tr>
<tr>
<td>BiLSTM baseline</td>
<td>Williams et al.</td>
<td>67.0%</td>
<td>67.6%</td>
<td>Pool, ProdDiff, SNLI</td>
</tr>
</tbody>
</table>

Table 2: Key statistics for the corpus broken down by genre, presented alongside figures from SNLI for comparison. The first five genres represent the matched section of the development and test sets, and the remaining five represent the mismatched section. The first three statistics shown are the number of examples in each genre. #Wds. Prem. is the mean token count among premise sentences. 'S' parses is the percentage of premises or hypotheses which the Stanford Parser labeled as full sentences rather than fragments. Agrmt. is the percent of individual annotator labels that match the assigned gold label used in evaluation. BiLSTM Acc. gives the test accuracy on the full test set for the BiLSTM baseline model trained on MultiNLI and SNLI.

Table 3: RepEval 2017 shared task competition results. The Model Details column lists some of the key strategies used in each system, using keywords: STACK: use of multilayer bidirectional RNNs, CHAR: character-level embeddings, ENH: embeddings enhanced with auxiliary features, POOL: max or mean pooling over RNN states, Attn: intra-sentence attention, ProdDiff: elementwise sentence product and difference features in the final entailment classifier, SNLI: use of the SNLI training set.

**Deeper Analysis**

- **Embeddings**: Systems vary reasonably widely in their approach to input encoding. Yang et al. and Chen et al. use a combination of GloVe embeddings (Pennington et al., 2014, not fine tuned) and character-level convolutional neural networks (Kim et al., 2016) to extract representations of words. Balazs et al. also use pre-trained GloVe embeddings without fine tuning, but report (contra Chen et al.) that an added character-level feature extractor does not improve performance. Vu et al. use pre-trained GloVe word embeddings augmented with additional feature vectors. They create embeddings for part-of-speech (POS), character level information, and the dependency relation between a word and its parent, and con-

---

[Nangia et al., 2017]
Entailment-based Embeddings

- Shortcut-stacked RNNs also achieved encoding-based SotA on SNLI corpus

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNLI</td>
</tr>
<tr>
<td>CBOW (Williams et al., 2017)</td>
<td>80.6</td>
</tr>
<tr>
<td>biLSTM Encoder (Williams et al., 2017)</td>
<td>81.5</td>
</tr>
<tr>
<td>300D Tree-CNN Encoder (Mou et al., 2015)</td>
<td>82.1</td>
</tr>
<tr>
<td>300D SPINN-PI Encoder (Bowman et al., 2016)</td>
<td>83.2</td>
</tr>
<tr>
<td>300D NSE Encoder (Munkhdalai and Yu, 2016)</td>
<td>84.6</td>
</tr>
<tr>
<td>biLSTM-Max Encoder (Conneau et al., 2017)</td>
<td>84.5</td>
</tr>
<tr>
<td>Our biLSTM-Max Encoder</td>
<td>85.2</td>
</tr>
<tr>
<td>Our Shortcut-Stacked Encoder</td>
<td><strong>86.1</strong></td>
</tr>
</tbody>
</table>

[Nie and Bansal, 2017]
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

![Image](https://via.placeholder.com/150)

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.

**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
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
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

- Parse trees of 11,855 sentences
- 215,154 phrases with labels
- Allows training and evaluating with compositional information
Better Models: Recursive Neural Tensor Network (RNTN)
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
## 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><strong>88.1</strong></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>

Our results are summarized in Table 2. As was the case with the convolutional neural network model, our results are on par with the state of the art on the sentiment classification subtask. This is particularly impressive when compared to the results of existing systems on the fine-grained classification subtask. The sentiment classifier was additionally regularized using dropout (Hinton et al., 2012). Notice that the strength of this regularization was determined during training with a learning rate of 0.05 and a minibatch size of 25. The model parameters were tuned for the semantic relatedness subtask (indeed, we achieve similar results on all 5-class sentiment classification tasks). This suggests that fine-tuning helps distinguish positive/negative sentiments from neutral ones.

Our single-layer LSTM models achieved comparable results to the Bidirectional LSTM on the fine-grained classification subtask. This demonstrates that sentence length becomes a limiting factor in the fine-grained classification task, in contrast to the minor gains observed on the binary classification subtask. This suggests that the state that needs to be retained by the network for the (unidirectional) LSTM on the fine-grained classification subtask is easily preserved by both the LSTM and Bidirectional LSTM models, whereas the fine-grained classification subtask requires more complex interactions between the input word representations and the hidden state of the LSTM unit.

For the similarity prediction network (Eqs. 15) we use a hidden layer of size 50. We compare two Tree-LSTM architectures for compositionality, the Child-Sum and Tree-LSTM (Sec. 3.2) on binarized constituency trees (Klein and Manning, 2003). The Bidirectional LSTM significantly outperforms the sequential LSTM variants.

## Sentiment Compositionality

For the similarity prediction network (Eqs. 15) we use a hidden layer of size 50. We compare two Tree-LSTM architectures for compositionality, the Child-Sum and Tree-LSTM (Sec. 3.2) on binarized constituency trees (Klein and Manning, 2003). The Bidirectional LSTM significantly outperforms the sequential LSTM variants.

---

Demos: [http://nlp.stanford.edu:8080/sen7ment/rntnDemo.html](http://nlp.stanford.edu:8080/sen7ment/rntnDemo.html)

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

- Sentence similarity
- Entailment classification
- Spam detection
- Document topic classification
- Others: humor, rumor, sarcasm detection, etc.

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