Natural Language Processing

Info 159/259
Lecture 9: Embeddings 2 (Feb 16, 2021)

David Bamman, UC Berkeley
Distributed representation

• Vector representation that encodes information about the distribution of contexts a word appears in

• Words that appear in similar contexts have similar representations (and similar meanings, by the distributional hypothesis).
Dense vectors from prediction

- Learning low-dimensional representations of words by framing a predicting task: using context to predict words in a surrounding window
- Transform this into a *supervised* prediction problem; similar to language modeling but we’re ignoring order within the context window
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.418</td>
<td>0.24968</td>
<td>-0.41242</td>
<td>0.1217</td>
<td>...</td>
<td>-0.17862</td>
</tr>
<tr>
<td>,</td>
<td>0.013441</td>
<td>0.23682</td>
<td>-0.16899</td>
<td>0.40951</td>
<td>...</td>
<td>-0.55641</td>
</tr>
<tr>
<td>.</td>
<td>0.15164</td>
<td>0.30177</td>
<td>-0.16763</td>
<td>0.17684</td>
<td>...</td>
<td>-0.31086</td>
</tr>
<tr>
<td>of</td>
<td>0.70853</td>
<td>0.57088</td>
<td>-0.4716</td>
<td>0.18048</td>
<td>...</td>
<td>-0.52393</td>
</tr>
<tr>
<td>to</td>
<td>0.68047</td>
<td>-0.039263</td>
<td>0.30186</td>
<td>-0.17792</td>
<td>...</td>
<td>0.13228</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>chanty</td>
<td>0.23204</td>
<td>0.025672</td>
<td>-0.70699</td>
<td>-0.04547</td>
<td>...</td>
<td>0.34108</td>
</tr>
<tr>
<td>kronik</td>
<td>-0.60921</td>
<td>-0.67218</td>
<td>0.23521</td>
<td>-0.11195</td>
<td>...</td>
<td>0.85632</td>
</tr>
<tr>
<td>rolonda</td>
<td>-0.51181</td>
<td>0.058706</td>
<td>1.0913</td>
<td>-0.55163</td>
<td>...</td>
<td>0.079711</td>
</tr>
<tr>
<td>zsombor</td>
<td>-0.75898</td>
<td>-0.47426</td>
<td>0.4737</td>
<td>0.7725</td>
<td>...</td>
<td>0.84014</td>
</tr>
<tr>
<td>sandberger</td>
<td>0.072617</td>
<td>-0.51393</td>
<td>0.4728</td>
<td>-0.52202</td>
<td>...</td>
<td>0.23096</td>
</tr>
</tbody>
</table>

https://nlp.stanford.edu/projects/glove/
Word embeddings

- Pre-trained word embeddings great for words that appear frequently in data
- Unseen words are treated as UNKs and assigned zero or random vectors; everything unseen is assigned the same representation.
- supercalifragilisticexpialidocious
  - super, superior, supernatural
  - adventurous, fabulous, infamous
- supercalifragilisticexpialidociously
  - quickly, sadly, perfectly
## Agglutinative languages

<table>
<thead>
<tr>
<th>Turkish Expression</th>
<th>English Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muvaffakiyetsizleş(-mek)</td>
<td>(To) become unsuccessful</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştir(-mek)</td>
<td>(To) make one unsuccessful</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştirici</td>
<td>Maker of unsuccessful ones</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştiricileş(-mek)</td>
<td>(To) become a maker of unsuccessful ones</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştiricileştir(-mek)</td>
<td>(To) make one a maker of unsuccessful ones</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştiricileştiriver(-mek)</td>
<td>(To) easily/quickly make one a maker of unsuccessful ones</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştiricileştiriverebil(-mek)</td>
<td>(To) be able to make one easily/quickly a maker of unsuccessful ones</td>
</tr>
<tr>
<td>Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsinizcesine</td>
<td>As though you happen to have been from among those whom we will not be able to easily/quickly make a maker of unsuccessful ones</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>danser</td>
<td>dansé</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

https://www.collinsdictionary.com/dictionary/french-english/conjugation/danser
Even in languages like English that are not agglutinative and aren’t highly inflected, words share important structure.

Even if we never see the word “unfriendly” in our data, we should be able to reason about it as: un + friend + ly
Subword models

• Rather than learning a single representation for each word type $w$, learn representations $z$ for the set of ngrams $\mathcal{G}_w$ that comprise it [Bojanowski et al. 2017]

• The word itself is included among the ngrams (no matter its length).

• A word representation is the sum of those ngrams

$$w = \sum_{g \in \mathcal{G}_w} z_g$$
FastText

\[ e(\text{where}) = \]

\[ e(\text{<wh}) + e(\text{whe}) + e(\text{her}) + e(\text{ere}) + e(\text{re>}) + e(\text{<whe}) + e(\text{wher}) + e(\text{here}) + e(\text{ere>}) + e(\text{<whe}) + e(\text{wher}) + e(\text{here}) + e(\text{ere>}) + e(\text{<where}) + e(\text{where}) + e(\text{here>}) + e(\text{<where}) + e(\text{where>}) + e(\text{where>}) + e(\text{where>}) + e(\text{where>}) \]

\[ e(*) = \text{embedding for *} \]

3-grams

4-grams

5-grams

6-grams

word
• Subword models need less data to get comparable performance.
How do we use word embeddings for document classification?
I loved the movie!
I loved the movie!

Iyyer et al. (2015), “Deep Unordered Composition Rivals Syntactic Methods for Text Classification” (ACL)
I loved the movie!
I loved the movie!
Attention

- Let's incorporate structure (and parameters) into a network that captures which elements in the input we should be attending to (and which we can ignore).
Define $v$ to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector.
\[ v \in \mathcal{R}^H \]

\[
\begin{align*}
    r_1 &= v^\top x_1 \\
    r_2 &= v^\top x_2 \\
    r_3 &= v^\top x_3 \\
    r_4 &= v^\top x_4 \\
    r_5 &= v^\top x_5
\end{align*}
\]

\[
\begin{array}{cccccc}
    2.7 & 3.1 & -1.4 & -2.3 & 0.7 \\
    -3.4 & 2.4 & -0.8 & -1.2 & 1.7 \\
\end{array}
\]

X1

X2

X3

X4

X5
Convert $r$ into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$

<table>
<thead>
<tr>
<th>$a$</th>
<th>0</th>
<th>0.64</th>
<th>0.02</th>
<th>0.02</th>
<th>0.32</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>-3.4</td>
<td>2.4</td>
<td>-0.8</td>
<td>-1.2</td>
<td>1.7</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
    r_1 &= v^T x_1 \\
    r_2 &= v^T x_2 \\
    r_3 &= v^T x_3 \\
    r_4 &= v^T x_4 \\
    r_5 &= v^T x_5
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>2.7</th>
<th>3.1</th>
<th>-1.4</th>
<th>-2.3</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.7</td>
<td>-0.8</td>
<td>-1.3</td>
<td>-0.2</td>
<td>-0.9</td>
</tr>
<tr>
<td></td>
<td>2.3</td>
<td>1.5</td>
<td>1.1</td>
<td>1.4</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>-0.9</td>
<td>-1.5</td>
<td>-0.7</td>
<td>0.9</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
<td>-0.7</td>
<td>-1.6</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

I loved the movie!
I loved the movie!

\[ y = x_1 a_1 + x_2 a_2 + x_3 a_3 + x_4 a_4 + x_5 a_5 \]

Weighted sum

<table>
<thead>
<tr>
<th>2.7</th>
<th>3.1</th>
<th>-1.4</th>
<th>-2.3</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.7</td>
<td>-0.8</td>
<td>-1.3</td>
<td>-0.2</td>
<td>-0.9</td>
</tr>
<tr>
<td>2.3</td>
<td>1.5</td>
<td>1.1</td>
<td>1.4</td>
<td>1.3</td>
</tr>
<tr>
<td>-0.9</td>
<td>-1.5</td>
<td>-0.7</td>
<td>0.9</td>
<td>0.2</td>
</tr>
<tr>
<td>-0.1</td>
<td>-0.7</td>
<td>-1.6</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

I loved the movie!
Attention

• Lots of variations on attention:
  • Linear transformation of x into before dotting with v
  • Non-linearities after each operation.
  • “Multi-head attention”: multiple v vectors to capture different phenomena that can be attended to in the input.
  • Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).
Attention

- Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction.
- This can be useful for visualization.
• With an RNN, we can generate a representation of the sequence as seen through time \( t \).

• This encodes a representation of meaning specific to the local context a word is used in.
We can then swap that RNN time step output for the embeddings we used earlier.
What about the future context?
Bidirectional RNN

- A powerful alternative is make predictions conditioning both on the past and the future.

- Two RNNs
  - One running left-to-right
  - One right-to-left

- Each produces an output vector at each time step, which we concatenate
I loved the movie!
I loved the movie!
Bidirectional RNN

I

loved

the

movie

I

loved

the

movie

!
Bidirectional RNN

- The forward RNN and backward RNN each output a vector of size $H$ at each time step, which we concatenate into a vector of size $2H$.

- The forward and backward RNN each have separate parameters to be learned during training.
Training BiRNNs

• Given this definition of an BiRNN:

\[ s^i_b = R_b(x^i, s^i_{b+1}) = g(s^i_{b+1} W^s_b + x^i W^x_b + b_b) \]
\[ s^i_f = R_f(x^i, s^i_{f-1}) = g(s^i_{f-1} W^s_f + x^i W^x_f + b_f) \]

\[ y_i = \text{softmax} \left( [s^i_f; s^i_b] W^o + b^o \right) \]

• We have 8 sets of parameters to learn (3 for each RNN + 2 for the final layer)
Stacked RNN

- Multiple RNNs, where the output of one layer becomes the input to the next.
Contextualized embeddings

• Models for learning static embeddings learn a single representation for a word type.
Types and tokens

- Type: bears

- Tokens:
  - The bears ate the honey
  - We spotted the bears from the highway
  - Yosemite has brown bears
  - The chicago bears didn’t make the playoffs

<table>
<thead>
<tr>
<th>&quot;bears&quot;</th>
<th>3.1</th>
<th>1.4</th>
<th>-2.7</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.1</td>
<td>1.4</td>
<td>-2.7</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>1.4</td>
<td>-2.7</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>1.4</td>
<td>-2.7</td>
<td>0.3</td>
</tr>
</tbody>
</table>
• Big idea: transform the representation of a token in a sentence (e.g., from a static word embedding) to be sensitive to its local context in a sentence and trainable to be optimized for a specific NLP task.
Stacked BiRNN trained to predict next word in language modeling task

Transformer-based model to predict masked word using bidirectional context + next sentence prediction.

ELMo

BERT

Peters et al. 2018

Devlin et al. 2019
ELMo


• Big idea: transform the representation of a word (e.g., from a static word embedding) to be sensitive to its local context in a sentence and optimized for a specific NLP task.

• Output = word representations that can be plugged into just about any architecture a word embedding can be used.
• Train a bidirectional RNN language model with L layers on a bunch of text.

• Learn parameters to combine the RNN output across all layers for each word in a sentence for a specific task (NER, semantic role labeling, question answering etc.). Large improvements over SOTA for lots of NLP problems.
Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F₁ for SQuAD, SRL and NER; average F₁ for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The “increase” column lists both the absolute and relative improvements over our baseline.
• Transformer-based model (Vaswani et al. 2017) to predict masked word using bidirectional context + next sentence prediction.

• Generates multiple layers of representations for each token sensitive to its context of use.
Each token in the input starts out represented by token and position embeddings

The

e1,1

-0.2 1 0.1 -0.8 -1.1

dog

e1,2

0.3 0.3 -1.7 0.7 -1.1

barked

e1,3

1.6 -0.3 -0.9 -0.7 0.2
The value for time step $j$ at layer $i$ is the result of attention over all time steps in the previous layer $i-1$.

\[
\begin{array}{ccccc}
-0.7 & -1.3 & 0.4 & -0.4 & -0.7 \\
\end{array}
\]

$e_{2,1}$

\[
\begin{array}{cccc}
-0.2 & 1 & 0.1 & -0.8 & -1.1 \\
\end{array}
\]

$e_{1,1}$

\[
\begin{array}{cccc}
0.3 & 0.3 & -1.7 & 0.7 & -1.1 \\
\end{array}
\]

$e_{1,2}$

\[
\begin{array}{cccc}
1.6 & -0.3 & -0.9 & -0.7 & 0.2 \\
\end{array}
\]

$e_{1,3}$

The dog barked
The dog barked.
The dog barked
The dog barked
The dog barked.
The dog barked
The dog barked
At the end of this process, we have one representation for each layer for each token.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.2</td>
<td>0.3</td>
<td>2.1</td>
<td>1.2</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{3,1}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.8</td>
<td>-0.2</td>
<td>-2.4</td>
<td>-0.2</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{3,2}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.9</td>
<td>-1.5</td>
<td>-0.7</td>
<td>0.9</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{3,3}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.7</td>
<td>-1.3</td>
<td>0.4</td>
<td>-0.4</td>
<td>-0.7</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{2,1}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>-1.1</td>
<td>1.1</td>
<td>0.6</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{2,2}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.1</td>
<td>-0.7</td>
<td>-0.1</td>
<td>0.9</td>
<td>-1.1</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{2,3}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.2</td>
<td>1</td>
<td>0.1</td>
<td>-0.8</td>
<td>-1.1</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{1,1}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>-1.7</td>
<td>0.7</td>
<td>-1.1</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{1,2}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>-0.3</td>
<td>-0.9</td>
<td>-0.7</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>\text{e}_{1,3}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**WordPiece**

- BERT uses WordPiece tokenization, which segments some morphological structure of tokens.
- Vocabulary size: 30,000

<table>
<thead>
<tr>
<th>The</th>
<th>The</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>dog</td>
</tr>
<tr>
<td>barked</td>
<td>bark #ed</td>
</tr>
</tbody>
</table>
• BERT also encodes each sentence by appending a special token to the beginning ([CLS]) and end ([SEP]) of each sequence.

• This helps provides a single token that can be optimized to represent the entire sequence (e.g., for document classification)
• We can represent the entire document with this *one* [CLS] vector
• Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).
neutral sentiment

- We can represent the entire document with this *one* [CLS] vector.
- Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).
BERT

• Learn the parameters of this model with two objectives:
  • Masked language modeling
  • Next sentence prediction
Masked LM

- Mask one word from the input and try to predict that word as the output.

- More powerful than an RNN LM (or even a BiRNN LM) since it can reason about context on both sides of the word being predicted.

- A BiRNN models context on both sides, but each RNN only has access to information from one direction.
The [MASKED] bark #ed [SEP]
The [MASKED] bark #ed
bark

[CLS] The dog [MASKED] #ed [SEP]
Next sentence prediction

• For a pair of sentences, predict from [CLS] representation whether they appeared sequentially in the training data:

  ✖️ [CLS] The dog bark #ed [SEP] Paris is in France
  ✔️ [CLS] The dog bark #ed [SEP] He was hungry
BERT

• Deep layers (12 for BERT base, 24 for BERT large)

• Large representation sizes (768 per layer)

• Pretrained on English Wikipedia (2.5B words) and BooksCorpus (800M words)
Yosemite has brown bears.

We saw a moose in Alaska.

Da bears lost again!

Go pack go!
Progress — Coreference resolution

- Classic
  - Lee et al. 2011
  - Björkelund and Farkas 2012
  - Durrett and Klein 2013

- Neural
  - Lee et al. 2017

- ELMO
  - Peters et al. 2018

- BERT
  - Joshi et al. 2019

Scores:
- Classic: 56.65, 60.6, 67.2
- Neural: 67.2
- ELMO: 70.4
- BERT: 76.9
Bertology

- Hewitt et al. 2019
- Tenney et al. 2019
- McCoy et al. 2019
- Liu et al. 2019
- Clark et al. 2019
- Goldberg 2019
- Michel et al. 2019

Code

Pre-trained models for BERT, Transformer-XL, ALBERT, RoBERTa, DistilBERT, GPT-2, etc. for English, French, “Multilingual”

https://huggingface.co
• Even though BERT is mainly trained on a language modeling objective, it learns a lot about the structure of language — even without direct training data for specific linguistic tasks.

• Probing experiments uncover what—and where (in what layers)—pretrained BERT encodes this information.

Tenney et al. (2019), "BERT RedisCOVERs the Classical NLP Pipeline"
• Word embeddings can be substituted for one-hot encodings in many models (MLP, CNN, RNN, logistic regression).

• Subword embeddings allow you to create embeddings for word not present in training data; require much less data to train.

• Attention gives us a mechanism to learn which parts of a sequence to pay attention to more in forming a representation of it.

• BiLSTMs can transform word embeddings to be sensitive to their use in context.

• Static word embeddings (word2vec, Glove) provide representations of word types; contextualized word representations (ELMo, BERT) provide representations of tokens in context.