Lecture 4
Convolutional Neural Networks

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
01.03.2022
Plan for today

- Feed-forward neural network for text classification
- CNN as an analog to human visual system
- CNN components
  - Convolutional layers
  - Pooling layers
  - Linear and non-linear layers
- An example CNN for text classification
- (Interpreting CNN decisions)
Feed-forward NN for text classification
Feed-forward sentence classification

\[
h = g(W^{(i)}x + b^{(i)})
\]
\[
z = W^{(o)}h + b^{(o)}
\]
\[
p(c|x) = \text{softmax}(z)
\]

Embedding layer can be initialized with the pretrained word embeddings.
Linear layer

- Linear layer is just a parameter matrix
- The input vector is multiplied with the linear layer matrix from the left

\[ y = x \cdot W \]
Bias vector

- Often a bias vector is part of the linear layer

\[ y = x \cdot W + b \]
Non-linear activation functions

- Typically applied on top of a linear layer
- Except for the linear output layer – after that typically comes the softmax
Convolutional neural networks

- Feature extractors
- Analogous to the brain’s visual system
- First applied to image processing domain
- Now (sometimes) also used in NLP
Feature extraction in image detection
Visual system in the brain
Convolutional NN for image classification
“Artificial” and “natural” CNN
Convolutional layers
Convolutional layers

\[ O_{0,0} = I_{0,0}F_{0,0} + I_{0,1}F_{0,1} + I_{0,2}F_{0,2} + I_{1,0}F_{1,0} + I_{1,1}F_{1,1} + I_{1,2}F_{1,2} + I_{2,0}F_{2,0} + I_{2,1}F_{2,1} + I_{2,2}F_{2,2} \]

\[ O_{4,1} = I_{4,1}F_{1,1} + I_{4,2}F_{1,2} + I_{4,3}F_{1,3} + I_{5,1}F_{2,1} + I_{5,2}F_{2,2} + I_{5,3}F_{2,3} + I_{6,1}F_{3,1} + I_{6,2}F_{3,2} + I_{6,3}F_{3,3} \]

\[ O_{i,j} = \sum_{m=0}^{k-1} \sum_{n=0}^{l-1} I_{i+m,j+n} \cdot F_{m,n} \]

Convolutional layer for text

The cat sat on the mat
Convolutional layer for text

- The convolutional filter is an n-gram detector
- Concatenate the embeddings of the n-gram
- Compute dot-product between the concatenated embedding vector and the filter
- The result is a scalar score for the n-gram
Convolutional layer for text

The cat sat on the mat
Convolutional layer for text

The cat sat on the mat

Embed Embed Embed Embed Embed Embed

The cat sat on the mat
Convolutional layer for text

The cat sat on the mat
Convolutional layer for text

The cat sat on the mat
Another (but equivalent) formulation

Element-wise multiplication \( \times \)

Then sum up

Again, apply to every n-gram
Different filters: unigrams, bigrams, trigrams etc

![Diagram showing unigrams, bigrams, and trigrams with 'The', 'cat', 'sat', 'on', 'the', 'mat' tokens and corresponding embeddings.

The cat sat on the mat.
Number of filters

- There are typically several filters for each n-gram type
- This is a tunable hyperparameter

The cat sat on the mat

Number of filters = 3
Final representation

- After applying 5 different filters to unigrams, bigrams and trigrams
Narrow and wide convolution

• Narrow convolution:
  • start the n-grams from the first word and end with the last word in the text
  • The resulting vector has n-k+1 elements
  • n – length of the sequence
  • k – n-gram length

The cat sat on the mat
Narrow and wide convolution

- **Wide convolution:**
  - Pad the sequence from the beginning and with a special PAD symbol
  - The length of the resulting vector is equal to $n + k - 1$

```
The cat sat on the mat
PAD PAD
```
Wide convolution

- Trigram filter
Pooling layers
Pooling layers

- The goal is to reduce the dimensionality of the feature map (output of the convolutional layer)
- Brings the representations of different inputs to the same dimensionality
- Typical pooling operations:
  - max-pooling (over time)
  - average pooling
Max-pooling over time

- Retain the maximum value over all values computed by a filter
- The information about the most relevant n-grams are kept

Number of filters = 3

The cat sat on the mat
Max-pooling over time

The cat sat on the mat
Max-pooling over time

The cat sat on the mat
The cat sat on the mat.
The cat sat on the mat
The cat sat on the mat.
Other pooling layers

- Average pooling: compute the average along the time axis
- K-max-pooling: retain k maximum elements (instead of just one), keep the original order of the elements
An example CNN for text classification
Y. Kim, 2014. Convolutional Neural Networks for Sentence Classification
Things to consider

- Pretrained word embeddings or learn from scratch?
- What kind of n-gram filters?
- How many filters per n-gram?
- Dropout?
- How to model the output layer in binary case? One-dimensional with sigmoid or two-dimensional with softmax?
- Something else?
When to consider using CNN for text classification?

- CNN can handle long texts well
- It’s computationally relatively cheap model (good for large datasets)
- Offers some interpretability
Interpreting CNNs
Decision interpretation

- For each test item it is possible to check which n-grams passed the max-pool and thus contributed to the classification decision.

### Examples from: Jacovi et al., 2018. Understanding Convolutional Neural Networks for Text Classification

<table>
<thead>
<tr>
<th>filter</th>
<th>f-class</th>
<th>ngram</th>
<th>slot scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>pos</td>
<td>PAD PAD my</td>
<td>0.7  1.65  0.16</td>
</tr>
<tr>
<td>1</td>
<td>pos</td>
<td>. very well</td>
<td>0.98 2.17  2.63</td>
</tr>
<tr>
<td>2</td>
<td>neg</td>
<td>PAD my UNK</td>
<td>1.31 -0.07  0.21</td>
</tr>
<tr>
<td>3</td>
<td>neg</td>
<td>UNK fits perfectly</td>
<td>0.28  0.61  0.03</td>
</tr>
<tr>
<td>4</td>
<td>neg</td>
<td>looking and offers</td>
<td>0.6  0.12  0.5</td>
</tr>
<tr>
<td>5</td>
<td>neg</td>
<td>good protection PAD</td>
<td>0.52  1.6  -0.01</td>
</tr>
<tr>
<td>6</td>
<td>pos</td>
<td>UNK fits perfectly</td>
<td>-0.06  2.36  1.82</td>
</tr>
<tr>
<td>7</td>
<td>neg</td>
<td>fits perfectly .</td>
<td>1.34 -0.71  1.47</td>
</tr>
<tr>
<td>8</td>
<td>neg</td>
<td>. very well</td>
<td>-0.01  1.97  -0.55</td>
</tr>
<tr>
<td>9</td>
<td>pos</td>
<td>perfectly . very</td>
<td>4.13  0.45  -0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>filter</th>
<th>f-class</th>
<th>ngram</th>
<th>slot scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>pos</td>
<td>product sucked was</td>
<td>0.12  2.05  0.1</td>
</tr>
<tr>
<td>1</td>
<td>pos</td>
<td>overall a bad</td>
<td>2.53  1.4  -1.16</td>
</tr>
<tr>
<td>2</td>
<td>neg</td>
<td>lights did n’t</td>
<td>-0.33  1.12  1.63</td>
</tr>
<tr>
<td>3</td>
<td>neg</td>
<td>PAD this product</td>
<td>-0.2  1.43  0.51</td>
</tr>
<tr>
<td>4</td>
<td>neg</td>
<td>did n’t work</td>
<td>1.21  0.97  2.65</td>
</tr>
<tr>
<td>5</td>
<td>neg</td>
<td>sucked was not</td>
<td>0.98  0.59  1.32</td>
</tr>
<tr>
<td>6</td>
<td>pos</td>
<td>work overall a</td>
<td>-0.25  4.05  -0.21</td>
</tr>
<tr>
<td>7</td>
<td>neg</td>
<td>was not loud</td>
<td>-0.33  2.85  0.52</td>
</tr>
<tr>
<td>8</td>
<td>neg</td>
<td>a bad product</td>
<td>-0.45  3.08  1.32</td>
</tr>
<tr>
<td>9</td>
<td>pos</td>
<td>PAD PAD this</td>
<td>0.38  0.15  1.66</td>
</tr>
</tbody>
</table>
Model interpretation

- Data from reddit
- Model for predicting users with self-reported eating disorder
- Posts extracted from non mental health related sub-reddits
- The model configuration
  - The model architecture of Kim (2014)
  - n-gram filters: unigrams to 5-grams
  - 100 filters per n-gram
  - Inputs: all posts per user concatenated together
  - Maximum length of the input: 15000 words
  - All words appearing less than 20 times are replaced with UNK token
- Look, which n-grams most frequently passed through the max-pool of each filter