Effective Use of Word Order for Text Categorization with Convolutional Neural Network

Presenter: Yi-Hsin Chen
Text Categorization

• Automatically assign pre-defined categories to documents written in natural language
  • Sentiment Classification
  • Topic Categorization
  • Spam Detection
Previous Works

• First representing a document using a bag-of-n-gram vector and then using SVM for classification
  • Lose information of word order

• First converting words to vectors as the input, then using Convolutional Neural Network (CNN) for classification
  • CNN output will retain the word order information
  • The word embedding might need separate training and additional resources
N-Gram

• A set of co-occurring words within a given window

• For example, given a sentence “How are you doing”
  • For N=2, there are three 2-gram: “How are”, “are you”, “you doing”
  • For N=3, there are two 3-gram: “How are you”, “are you doing”
Convolutional Neural Network (1/2)

- Convolution Layer
  - The output will retain the location information
  - Usually the input is a 3-D matrix (Height x Width x Channel) rather than a 2-D one
  - Followed by a non-linear activation function, ex: \( \text{ReLU} = \max(0, x) \)
  - Key Parameters:
    - Kernel size
    - Stride / Padding
    - # of Kernel
Convolutional Neural Network (2/2)

- **Pooling Layer**
  - Pooling down-samples the input spatially
  - The pooling function could be any function you want, the two most common ones are: 1) Max Pooling 2) Average Pooling
  - Key Parameters:
    - Kernel Size
    - Stride / Padding

![Kernel: 2x2 Stride: 2]

<table>
<thead>
<tr>
<th>Kernel Size</th>
<th>Stride / Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Max Pooling

<table>
<thead>
<tr>
<th>Avg. Pooling</th>
<th>Max Pooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
View Sentences as Images

- View each word as a “pixel” of an image

**Words**

<table>
<thead>
<tr>
<th></th>
<th>Hi,</th>
<th>how</th>
<th>are</th>
<th>you</th>
<th>doing?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**One-Hot Vectors**

- V: # of words in vocabulary
- N: # of words in the sentence

**Stack Vectors to an “image”**

1 x N x V “Image”

**Apply CNN**

1 x p kernel
Proposed Models

• Directly apply CNN to learn the embedding of a text region

• Seq-CNN: treat each word as an entity
  • For a 1 x p kernel, there will be p x V parameters
  • Harder to train, easier to overfit

• Bow-CNN: treat p words as an entity
  • Reduce # of parameter from p x V to V
  • Lose the order information for these p words

• Parallel-CNN: use multiple CNNs in parallel to learn multiple types of embedding to improve performance
Seq-CNN v.s. Bow-CNN

Words

Hi, how are you doing?

One-Hot Vectors

Seq-CNN

\[
\begin{pmatrix}
0 \\
0 \\
1 \\
0 \\
0
\end{pmatrix}
\quad \begin{pmatrix}
0 \\
1 \\
0 \\
0 \\
0
\end{pmatrix}
\quad \begin{pmatrix}
0 \\
0 \\
0 \\
1 \\
1
\end{pmatrix}
\quad \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}
\quad \begin{pmatrix}
1 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}
\]

Bow-CNN

\[
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}^T
\]

\[
\begin{pmatrix}
0 & 1 & 1 & 0 & 0
\end{pmatrix}^T
\]
Experiment

• Dataset
  • IMDB: movie review (Sentiment Classification)
  • Elec: electronics product reviews (Sentiment Classification)
  • RCV1 (topic categorization)

• Performance Benchmark (Error Rate)
  • The proposed models outperform B/L
  • The model configuration for sentiment classification and topic categorization is quite different

<table>
<thead>
<tr>
<th>methods</th>
<th>IMDB</th>
<th>Elec</th>
<th>RCV1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM bow3 (30K)</td>
<td>10.14</td>
<td>9.16</td>
<td>10.68</td>
</tr>
<tr>
<td>SVM bow1 (all)</td>
<td>11.36</td>
<td>11.71</td>
<td>10.76</td>
</tr>
<tr>
<td>SVM bow2 (all)</td>
<td>9.74</td>
<td>9.05</td>
<td>10.59</td>
</tr>
<tr>
<td>SVM bow3 (all)</td>
<td>9.42</td>
<td>8.71</td>
<td>10.69</td>
</tr>
<tr>
<td>NN bow3 (all)</td>
<td>9.17</td>
<td>8.48</td>
<td>10.67</td>
</tr>
<tr>
<td>NB-LM bow3 (all)</td>
<td>8.13</td>
<td>8.11</td>
<td>13.97</td>
</tr>
<tr>
<td>bow-CNN</td>
<td>8.66</td>
<td>8.39</td>
<td>9.33</td>
</tr>
<tr>
<td>seq-CNN</td>
<td>8.39</td>
<td>7.64</td>
<td>9.96</td>
</tr>
<tr>
<td>seq2-CNN</td>
<td>8.04</td>
<td>7.48</td>
<td>–</td>
</tr>
<tr>
<td>seq2-bown-CNN</td>
<td>7.67</td>
<td>7.14</td>
<td>–</td>
</tr>
</tbody>
</table>
Model Configuration for Different Tasks

• Sentiment Classification: a short phrase that conveys strong sentiment will dominate the results
  • Kernel size is small: 2~4
  • Using global max pooling

• Topic Categorization: need more context to provide information, the entire document matters, the location of text also matters
  • Kernel size is large: (20 for RCV1)
  • Using average pooling with 10 pooling units
CNN v.s. Bag-of-n-gram SVM (1/2)

• By directly learning the embedding of n-gram (n is decided by the kernel size), CNN is more able to utilize higher order n-gram for prediction

<table>
<thead>
<tr>
<th>Model</th>
<th>CNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Works perfectly! ,love this product</td>
<td>Great, excellent, perfect,</td>
</tr>
<tr>
<td></td>
<td>Very pleased! I am pleased</td>
<td>love, easy, amazing...</td>
</tr>
<tr>
<td>Negative</td>
<td>Completely useless., return policy</td>
<td>Poor, useless, returned,</td>
</tr>
<tr>
<td></td>
<td>It won’t even, but doesn’t work</td>
<td>not worth, return...</td>
</tr>
</tbody>
</table>

Predictive text region in the training set of Elec. dataset
CNN v.s. Bag-of-n-gram SVM (2/2)

- With the bag-of-n-gram representation, only the n-grams that appear in training data could help prediction.
- For CNN, even a n-gram doesn’t appear in the training data, once its constituent words does, it could still be helpful for prediction.

<table>
<thead>
<tr>
<th>Model</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Best concept ever, best idea ever, best hub ever, am wholly satisfied…</td>
</tr>
<tr>
<td>Negative</td>
<td>Were unacceptably bad, is abysmally bad, were universally poor…</td>
</tr>
</tbody>
</table>

Predictive text regions in the testing set which don’t appear in the training set.
Thank You For Your Attention!!!