Feature Projection for Improved Text Classification

Qi Qin, Wenpeng Hu, Bing Liu
ACL2020
Overview

• Many neural networks and embedding techniques have been designed to produce good representations for text classification (RNN, CNN, Transformer, BERT).

• **Indiscriminative** features encoded in representations could lead to **sub-optimal** classification performance.

• The goal of this work is to **project** existing representations produced by an encoder to a space where indiscriminative features are **eliminated**.
Feature Projection: Main Idea

• Given a data input, suppose that we can extract from it \textit{indiscriminative} features which are encoded in the vector $f_c$. (1)
• Using a regular encoder (e.g., RNN, CNN), we can extract features from the data input to store in $f_p$.
• We can factorize $f_p = f_p^* + \tilde{f}_p$ where $\tilde{f}_p \perp f_c$ and $f_p^* \parallel f_c$ (2)
  => using $\tilde{f}_p$ with all indiscriminative features \textit{eliminated} can be \textit{better} for classification.
• To achieve (1), they utilizes (\textit{in a new way}) the Gradient Reverse Layer (GRL).
• To achieve (2), they propose the Orthogonal Projection Layer (OPL).
Feature Projection: Overall Architecture
Feature Projection: C-Net

- **Inherit** the idea from domain adaptation field in using the Gradient Reverse Layer.
- Given an input sentence $X$, the feature vector extracted by CNN is: $f_c = \text{CNN}_c(X)$
- In forward pass, GRL serves as an identity function: $\text{GRL}(f_c) = f_c$
- Then, $f_c$ is used to predict the task label of the sentence: $Y_{GRL} = \text{softmax}(f_c \cdot W_c + b_c)$
- The regular cross-entropy loss is used here: $L_{oss_c} = \text{CrossEntropy}(Y_{truth}, Y_{GRL})$
- However, in backward pass, GRL interestingly reverse the direction of the gradient:
  \[
  \text{GRL}(x) = x, \\
  \frac{\partial \text{GRL}(x)}{\partial x} = -\lambda I,
  \]

=> This makes the updates for the feature extractor of C-Net are made toward **maximizing** the loss **instead** of minimizing it.

=> This is why $f_c$ becomes more and more **helpless** (i.e., **indiscriminative**) for classifying the sentence.
Feature Projection: C-Net

- In **domain adaptation**, the feature extractor is **shared** between the domain classifier and the task classifier.
Feature Projection: P-Net

- In P-Net, we also get a feature vector: \( f_p = \text{CNN}_p(X) \)
- With the indiscriminative feature vector \( f_c \) from C-Net, P-Net does some projections for \( f_p \) as follows:
  + First, \( f_p \) is \textbf{projected} on to \( f_c \) to obtain \( f_p^* : f_p^* = \text{Proj}(f_p, f_c) \)
  + Second, the \textbf{purified} feature vector \( \tilde{f}_p \) is computed by: \( \tilde{f}_p = f_p - f_p^* \)

- The purified vector \( \tilde{f}_p \) is then used for classification: \( Y_{OPL} = \text{softmax}(\tilde{f}_p \cdot W_p + b_p) \)
- To train the P-Net, the cross-entropy loss is used: \( \text{Loss}_p = \text{CrossEntropy}(Y_{\text{truth}}, Y_{OPL}) \)
- Not only can \( \tilde{f}_p \) benefit from \( f_c \) to be \textbf{more discriminative}, but also \( f_c \) can benefit from the discriminative signals of \( \tilde{f}_p \) to make \( f_c \) \textbf{more indiscriminative} (because \( \tilde{f}_p \perp f_c \)).
- P-Net and C-Net are \textbf{trained alternatively}, not jointly as in domain adaptation (in domain adaptation, we actually add two losses together).
Results

• Datasets:

| Data | $c$ | $l$ | $Train$ | $Test$ | $|V|$ |
|------|-----|-----|---------|--------|------|
| MR   | 2   | 45  | 8,529   | 1,066  | 17,884 |
| SNLI | 3   | 40  | 54,936  | 9,824  | 33,944 |
| SST2 | 2   | 35  | 6,920   | 1,821  | 16,789 |
| TREC | 6   | 15  | 5,000   | 952    | 8,834  |

Table 1: Dataset statistics. $c$: number of classes. $l$: average length of sentences, after padding and cutting. $Train$, $Test$: number of training and testing examples. $|V|$: vocabulary size.
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>SNLI</th>
<th>SST2</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>77.46(±0.41)</td>
<td>76.98(±0.07)</td>
<td>80.41(±0.20)</td>
<td>87.19(±0.58)</td>
</tr>
<tr>
<td>FP+LSTM</td>
<td>78.13(±0.18)</td>
<td>77.92(±0.10)</td>
<td>81.60(±0.17)</td>
<td>88.83(±0.40)</td>
</tr>
<tr>
<td>CNN</td>
<td>76.18(±0.45)</td>
<td>72.92(±0.19)</td>
<td>80.47(±0.59)</td>
<td>90.86(±0.51)</td>
</tr>
<tr>
<td>FP+CNN</td>
<td>78.74(±0.36)</td>
<td>74.38(±0.14)</td>
<td>82.02(±0.11)</td>
<td>92.78(±0.26)</td>
</tr>
<tr>
<td>Trans</td>
<td>75.18(±0.57)</td>
<td>66.71(±0.58)</td>
<td>76.93(±0.39)</td>
<td>87.33(±0.23)</td>
</tr>
<tr>
<td>FP+Trans</td>
<td>76.83(±0.66)</td>
<td>73.34(±0.43)</td>
<td>78.42(±0.49)</td>
<td>89.51(±0.79)</td>
</tr>
<tr>
<td>Bert</td>
<td>87.45(±0.51)</td>
<td>80.78(±0.42)</td>
<td>90.38(±0.10)</td>
<td>96.67(±0.22)</td>
</tr>
<tr>
<td>FP+Bert</td>
<td>90.56(±0.35)</td>
<td>81.47(±0.26)</td>
<td>92.24(±0.29)</td>
<td>98.33(±0.24)</td>
</tr>
</tbody>
</table>

Table 2: Results of our FP-Net against baseline methods. In each block, FP+X is a model obtained by our FP-Net using X as the feature extractor. Accuracy (%) is the evaluation metric. Each result in the table is the average accuracy of five experiments with the standard deviation in parentheses.