Feature Projection for Improved Text Classification

Qi Qin, Wenpeng Hu, Bing Liu

ACL2020

Overview

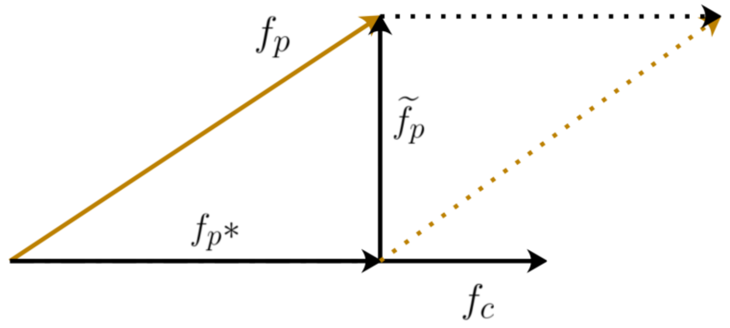
- Many neural networks and embedding techniques have been designed to produce **good** representations for text classification (RNN, CNN, Transformer, BERT).
- **Indiscriminative** features encoded in representations could lead to **sub-optimal** classification performance.
- The goal of this work is to **project** existing representations produced by an encoder to a space where indiscriminative features are **eliminated**.

Feature Projection: Main Idea

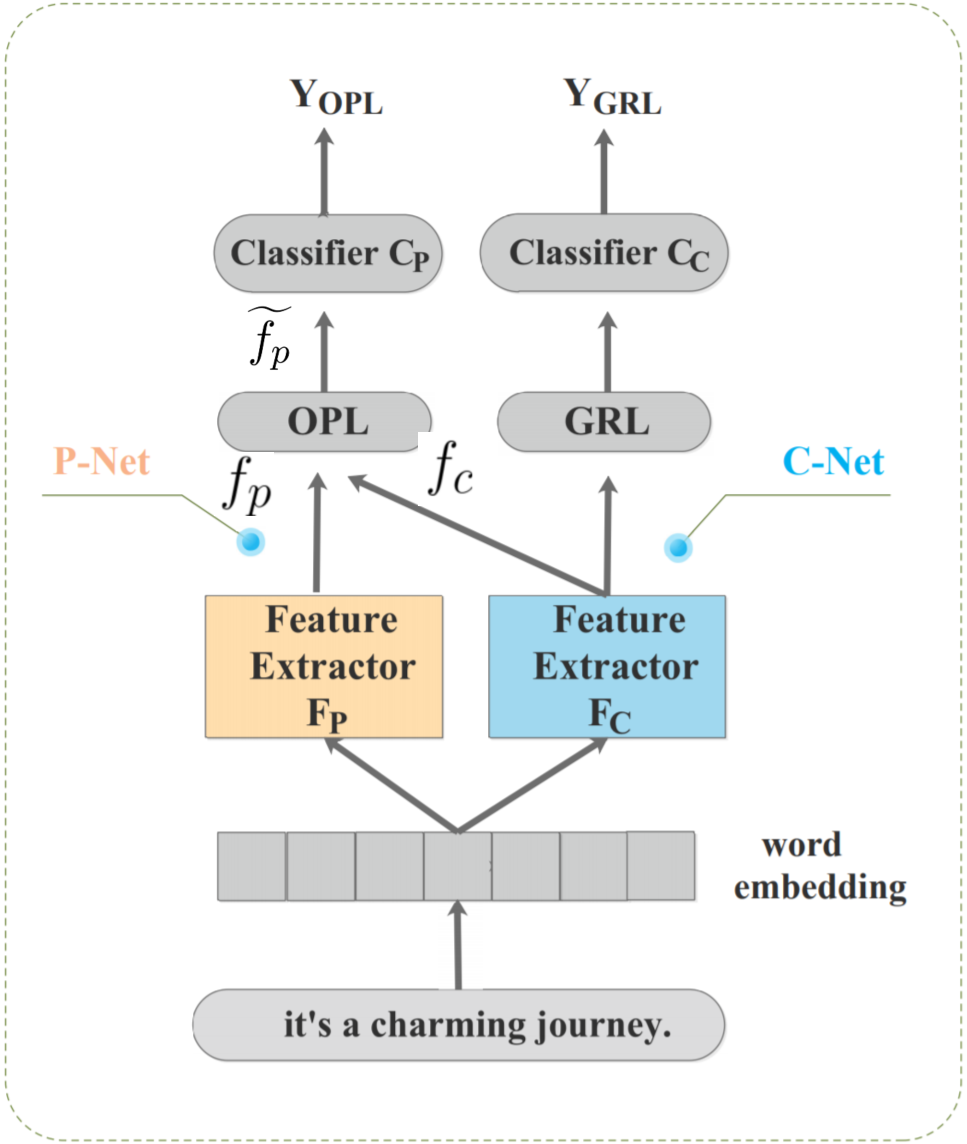
- Given a data input, suppose that we can extract from it **indiscriminative** features which are encoded in the vector f_c . (1)
- Using a regular encoder (e.g., RNN, CNN), we can extract features from the data input to store in f_p .
- We can factorize $f_p = f_p* + \widetilde{f_p}$ where $\widetilde{f_p} \perp f_c$ **and** $f_p* \parallel f_c$ (2)

=> using $\widetilde{f_p}$ with all indiscriminative features **eliminated** can be **better** for classification.

- To achieve (1), they utilizes (**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 adaptaion field in using the Gradient Reverse Layer.
- Given an input sentence X, the feature vector extracted by CNN is: $f_c = \mathrm{CNN}_\mathbf{C}(X)$
- In forward pass, GRL serves as an identity function: $GRL(f_c) = f_c$
- Then, f_c is used to predict the **task label** of the sentence: $Y_{GRL} = \mathrm{softmax}(f_c \cdot W_c + b_c)$
- The regular cross-entropy loss is used here: $Loss_c = \mathrm{CrossEntropy}(Y_{truth}, Y_{GRL})$
- However, in backward pass, GRL interestingly **reverse** the **direction** of the gradient:

$$GRL(x) = x,$$

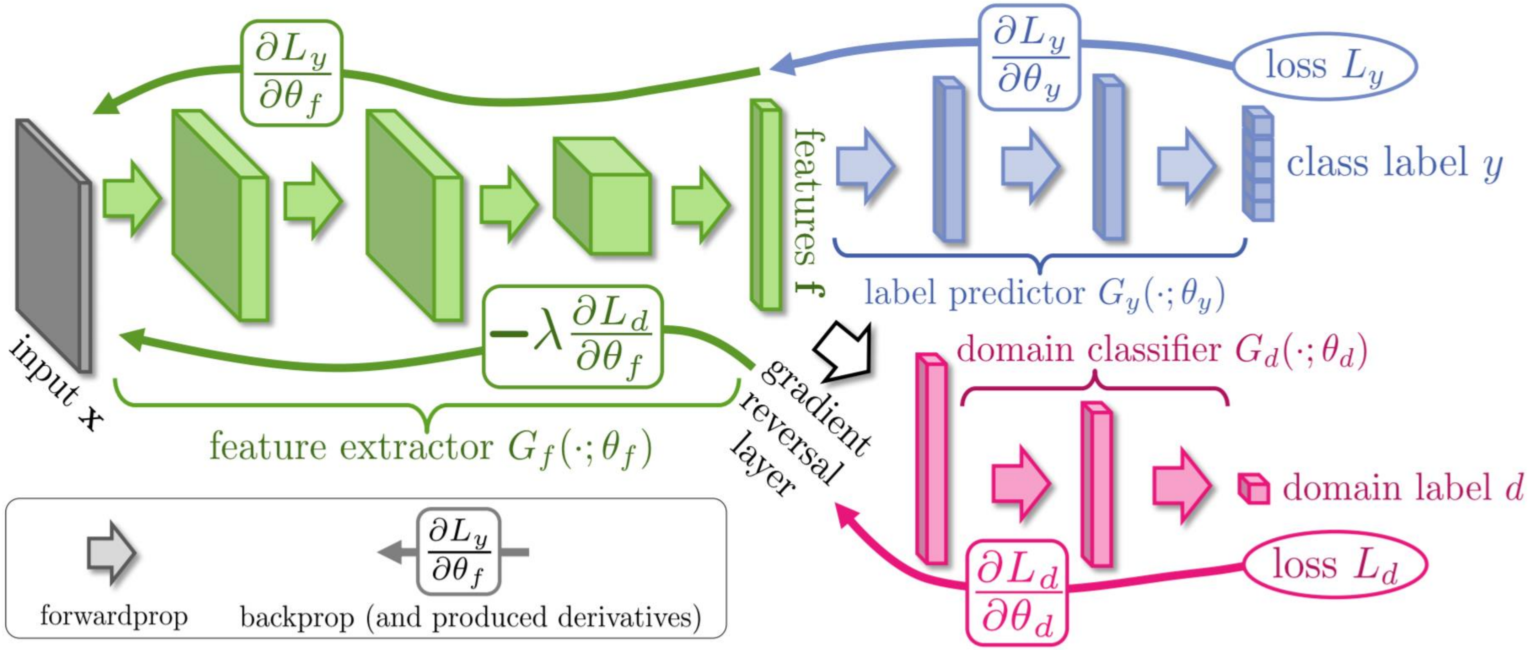
$$\frac{\partial GRL(x)}{\partial x} = -\lambda I,$$

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

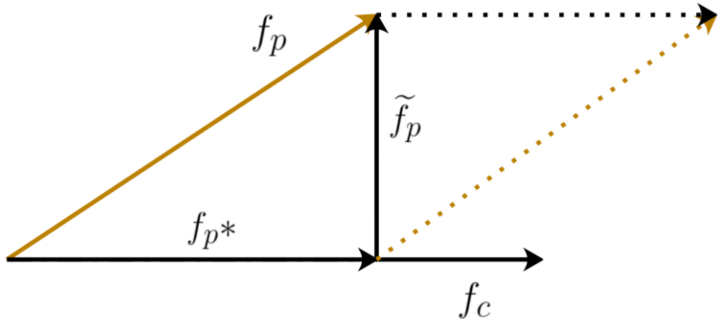
=> This is why f_c becomes more and more **helpless (i.e., indiscriminative)** for classifying the sentence.

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: $f_p = \mathrm{CNN_p}(X)$
- With the indiscriminative feature vector f_c from C-Net, P-Net does some projections for f_p as follows:

  + First, f_p is **projected** on to f_c to obtain f_p* : $f_p* = \mathrm{Proj}(f_p, f_c)$

  + Second, the **purified** feature vector $\widetilde{f_p}$ is computed by: $\widetilde{f_p} = f_p - f_p*$

- The purified vector $\widetilde{f_p}$ is then used for classification: $Y_{OPL} = \mathrm{softmax}(\widetilde{f_p} \cdot W_p + b_p)$
- To train the P-Net, the cross-entropy loss is used: $Loss_p = \mathrm{CrossEntropy}(Y_{truth}, Y_{OPL})$
- Not only can $\widetilde{f_p}$ benefit from f_c to be **more discriminative**, but also f_c can benefit from the discriminative signals of $\widetilde{f_p}$ to make f_c **more indiscriminative** (because $\widetilde{f_p} \perp f_c$).
- P-Net and C-Net are **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

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

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.