Algorithms and Software for Text Classification

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at Bloomberg, November 14, 2022
Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
Text Classification

- Examples
  C24 CCAT<TAB>uruguay uruguay compan ...
  C151 C15 CCAT<TAB>spun stak compan ...
- Binary/multi-class: each document is associated with exact one label
- Multi-label: each document is associated with multiple (maybe zero) labels
- This area has been well studied
Text Classification (Cont’d)

- However, we find that tools to **easily and conveniently** solve users’ problems are somewhat lacking.
- For example, among the various multi-label evaluation criteria such as Micro-F1, Macro-F1, Precision@$K$, NDCG@$K$, etc., which one should be used?
- Situations for data with few labels (e.g., \( \leq 1,000 \)) may be very different from those with millions of labels.
Can we guide users to solve their problems?

This is indeed difficult:

- No definitive recipes are available
- Things are a bit beyond what current autoML can do
The Project LibMultiLabel

- This is an on-going development for text classification
  
  https://www.csie.ntu.edu.tw/~cjlin/libmultilabel

- It is a simple tool with the following functionalities.
  - end-to-end services from raw texts to final evaluation/analysis
  - support for common neural network architectures and linear classifiers
  - easy hyper-parameter selection
The Project LibMultiLabel (Cont’d)

- We support
  - Binary/multi-class classification
  - Multi-label classification
- However, we haven’t had a good recipe yet for guiding users to effectively solve all their problems.
- In our on-going efforts for achieving this goal, we find that the inappropriate use of machine learning methods is now a big concern.
- We will share some interesting stories
Outline

1. Text classification and the project LibMultiLabel
2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models
3. Issues in designing a text classification package
   - Retraining after hyper-parameter search
4. Conclusions
Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
Parameter Selection in Machine Learning

- Everyone knows that hyper-parameter selection is important
- But in practice people may not pay enough attention
- In Liu et al. (2021), through an intriguing example we showed that even minor inattention can cause illusive research progress
Multi-label Classification for Medical Code Prediction

- MIMIC-III-full (Johnson et al., 2016): a multi-label set with 8,922 labels
  It is the most widely used open medical data set
- MIMIC-III-50: people follow Shi et al. (2017) to check the 50 most frequently occurring labels
- CAML (Mullenbach et al., 2018): an influential deep-learning work achieving state-of-the-art results on MIMIC-III-full and MIMIC-III-50
Subsequent Progress on MIMIC-III-50

Many subsequent works compared with CAML as a baseline and claimed SOTA results on the same MIMIC-III-50 set.
Did They Really Make Progress on MIMIC-III-50?

- How parameters were selected for CAML?
- MIMIC-III-full: Mullenbach et al. (2018) carefully tuned hyper-parameters by a validation process
- MIMIC-III-50: Mullenbach et al. (2018) directly used the parameters selected for MIMIC-III-full
- Usually we may think it’s not a big deal. But ...
Why inappropriate machine learning use is a concern

Story: inattention of parameter selection

Results of MIMIC-III-50 after Parameter Selection

- CAML by Mullenbach et al. (2018) is much better if parameters are selected
- Most subsequent developments cannot surpass the results
Discussion

- This example is intriguing because computational resources are not a concern
- Mullenbach et al. (2018) can do a search on the full data, so they can of course handle the top 50
- Sometimes a minor mis-step has a profound effect
- How can we avoid this?
Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
Why inappropriate machine learning use is a concern

Story: casual use of advanced models

How Most People Do Text Classification Now?

- BERT (Devlin et al., 2019), a large pre-trained model, has revolutionized many topics in NLP
- Due to its superior performance, people often take BERT and run a fixed number of epochs
- Such an advanced technique is great, but we will show that a casual use can sometimes be catastrophic.
- We give an illustration by considering the work by Chalkidis et al. (2022)
Results in Chalkidis et al. (2022)

- Chalkidis et al. (2022) released LexGLUE, a collection of legal-document data sets.
- Data sets are either multi-class or multi-label (# labels is small, ≤ 100).
- Both BERT-based methods and linear SVMs are included for their evaluation.
- They report two tables on performance and time.
Why inappropriate machine learning use is a concern

Story: casual use of advanced models

Results in Chalkidis et al. (2022):
Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
<th>UNFAIR-ToS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu$-F$_1$</td>
<td>m-F$_1$</td>
<td>$\mu$-F$_1$</td>
<td>m-F$_1$</td>
<td>$\mu$-F$_1$</td>
<td>m-F$_1$</td>
</tr>
<tr>
<td>TF-IDF+SVM</td>
<td>64.5</td>
<td>51.7</td>
<td>74.6</td>
<td>65.1</td>
<td>78.2</td>
<td>69.5</td>
</tr>
<tr>
<td>BERT</td>
<td>71.2</td>
<td>63.6</td>
<td>79.7</td>
<td>73.4</td>
<td>68.3</td>
<td>58.3</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>69.2</td>
<td>59.0</td>
<td>77.3</td>
<td>68.9</td>
<td>71.6</td>
<td>62.0</td>
</tr>
<tr>
<td>DeBERTa</td>
<td>70.0</td>
<td>60.8</td>
<td>78.8</td>
<td>71.0</td>
<td>71.1</td>
<td>62.7</td>
</tr>
<tr>
<td>Longformer</td>
<td>69.9</td>
<td>64.7</td>
<td>79.4</td>
<td>71.7</td>
<td>72.9</td>
<td>64.0</td>
</tr>
<tr>
<td>BigBird</td>
<td>70.0</td>
<td>62.9</td>
<td>78.8</td>
<td>70.9</td>
<td>72.8</td>
<td>62.0</td>
</tr>
<tr>
<td>Legal-BERT</td>
<td>70.0</td>
<td>64.0</td>
<td>80.4</td>
<td>74.7</td>
<td>76.4</td>
<td>66.5</td>
</tr>
<tr>
<td>CaseLaw-BERT</td>
<td>69.8</td>
<td>62.9</td>
<td>78.8</td>
<td>70.3</td>
<td>76.6</td>
<td>65.9</td>
</tr>
</tbody>
</table>

- $\mu$-F$_1$: Micro-F1; m-F$_1$: Macro-F1
- Chalkidis et al. (2022) have “TF-IDF+SVM” – this means linear but not kernel SVM
- SVM performs very well, especially for the last four problems
Why inappropriate machine learning use is a concern

Story: casual use of advanced models

Results in Chalkidis et al. (2022): Time

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>3h 42m</td>
<td>3h 9m</td>
<td>1h 24m</td>
<td>3h 36m</td>
<td>6h 9m</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>4h 11m</td>
<td>3h 43m</td>
<td>2h 46m</td>
<td>3h 36m</td>
<td>6h 22m</td>
</tr>
<tr>
<td>DeBERTa</td>
<td>7h 43m</td>
<td>6h 48m</td>
<td>3h 42m</td>
<td>5h 34m</td>
<td>9h 29m</td>
</tr>
<tr>
<td>Longformer</td>
<td>6h 47m</td>
<td>7h 31m</td>
<td>6h 27m</td>
<td>11h 10m</td>
<td>15h 47m</td>
</tr>
<tr>
<td>BigBird</td>
<td>8h 41m</td>
<td>8h 17m</td>
<td>5h 51m</td>
<td>3h 57m</td>
<td>8h 13m</td>
</tr>
<tr>
<td>Legal-BERT</td>
<td>3h 52m</td>
<td>3h 2m</td>
<td>2h 2m</td>
<td>3h 22m</td>
<td>5h 23m</td>
</tr>
<tr>
<td>CaseLaw-BERT</td>
<td>3h 2m</td>
<td>2h 57m</td>
<td>2h 34m</td>
<td>3h 40m</td>
<td>6h 8m</td>
</tr>
</tbody>
</table>

- This is **GPU** time
- But given linear SVM’s decent performance, why training time of linear SVM was not shown?
- We decide to have some investigation by using solvers in LibMultiLabel
In LibMultiLabel, three linear methods are provided
- Linear SVM and logistic regression (LR)
- Thresholding (Yang, 2001; Lewis et al., 2004; Fan and Lin, 2007): an extension of linear SVM/LR to optimize Macro-F1
- Cost-sensitive learning (Parambath et al., 2014): an extension of linear SVM/LR to optimize Micro-F1 or Macro-F1

All these techniques were developed long time ago
They basically need no parameter tuning, so let’s directly run them
Direct Run of BERT

- LibMultiLabel also supports BERT
- We check results without/with hyper-parameter selection
- Hyper-parameter search space

\[
\begin{array}{ccc}
\text{max_seq_length} & \text{learning_rate} & \text{dropout} \\
[128, 512] & [2e-5, 3e-5, 5e-5] & [0.1, 0.2] \\
\end{array}
\]
## Performance Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A) μ-F₁ m-F₁</th>
<th>ECtHR (B) μ-F₁ m-F₁</th>
<th>SCOTUS μ-F₁ m-F₁</th>
<th>EUR-LEX μ-F₁ m-F₁</th>
<th>LEDGAR μ-F₁ m-F₁</th>
<th>UNFAIR-ToS μ-F₁ m-F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM thresholding</td>
<td>64.0 53.1</td>
<td>72.8 63.9</td>
<td>78.1 68.9</td>
<td>72.0 55.4</td>
<td>86.4 80.0</td>
<td>94.9 75.1</td>
</tr>
<tr>
<td>SVM cost-sensitive</td>
<td>68.6 64.9</td>
<td>76.1 68.7</td>
<td>78.9 71.5</td>
<td>74.7 62.7</td>
<td>86.2 79.9</td>
<td>95.1 79.9</td>
</tr>
<tr>
<td>BERT</td>
<td>67.4 60.5</td>
<td>75.5 67.3</td>
<td>78.3 71.5</td>
<td>73.4 60.5</td>
<td>86.2 80.1</td>
<td>95.3 77.9</td>
</tr>
<tr>
<td>Our default</td>
<td>60.5 53.4</td>
<td>68.9 60.8</td>
<td>66.3 54.8</td>
<td>70.8 55.3</td>
<td>85.2 77.9</td>
<td>95.2 78.2</td>
</tr>
<tr>
<td>Our tuned</td>
<td>61.9 55.6</td>
<td>69.8 60.5</td>
<td>67.1 55.9</td>
<td>70.8 55.3</td>
<td>87.0 80.7</td>
<td>95.4 80.3</td>
</tr>
<tr>
<td>Chalkidis et al.</td>
<td>71.2 63.6</td>
<td>79.7 73.4</td>
<td>68.3 58.3</td>
<td>71.4 57.2</td>
<td>87.6 81.8</td>
<td>95.6 81.3</td>
</tr>
</tbody>
</table>

- A direct run of SVM is already close to BERT
- The two extensions (thresholding and cost-sensitive learning) are even more competitive
- However, for the first two problems, results of running LibMultiLabel’s BERT are poor even after hyper-parameter selection
Performance Comparison (Cont’d)

- We found that for the first three problems Chalkidis et al. (2022) used some sophisticated settings to run BERT
- For some documents, the average length is long

<table>
<thead>
<tr>
<th></th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
<th>UNFAIR-ToS</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>1,662.08</td>
<td>1,662.08</td>
<td>6,859.87</td>
<td>1,203.92</td>
<td>112.98</td>
<td>32.70</td>
</tr>
</tbody>
</table>

- But many may not know that BERT takes up to only 512 tokens
Chalkidis et al. (2022) split each long instance into 64 segments where each segment contains at most 128 tokens.

Each segment was fed into BERT, and [CLS] tokens were collected and input into an upper-level transformer.

The problem is that very often engineers directly run BERT without checking the document length.
Performance Comparison: Some Notes

- For each problem, training, validation, and test sets are available.
- What we showed are test performance, independent from training.
- For linear methods, training and validation sets are combined as cross validation may be internally done.
- For BERT, validation sets are used for selecting the best epoch and/or the best hyper-parameters.
- The model achieving the best validation performance is deployed for prediction.
Why inappropriate machine learning use is a concern  

Story: casual use of advanced models

Performance Comparison: Some Notes

- We tried to re-train the training and validation sets together. Results are improved in some cases.

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
<th>UNFAIR-ToS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>µ-F₁</td>
<td>m-F₁</td>
<td>µ-F₁</td>
<td>m-F₁</td>
<td>µ-F₁</td>
<td>m-F₁</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>64.0 53.1</td>
<td>72.8 63.9</td>
<td>78.1</td>
<td>68.9</td>
<td>72.0</td>
<td>55.4</td>
</tr>
<tr>
<td>thresholding</td>
<td>68.6 64.9</td>
<td>76.1 68.7</td>
<td>78.9</td>
<td>71.5</td>
<td>74.7</td>
<td>62.7</td>
</tr>
<tr>
<td>cost-sensitive</td>
<td>67.4 60.5</td>
<td>75.5 67.3</td>
<td>78.3</td>
<td>71.5</td>
<td>73.4</td>
<td>60.5</td>
</tr>
<tr>
<td>BERT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our default</td>
<td>60.5 53.4</td>
<td>68.9 60.8</td>
<td>66.3</td>
<td>54.8</td>
<td>70.8</td>
<td>55.3</td>
</tr>
<tr>
<td>Our tuned</td>
<td>61.9 55.6</td>
<td>69.8 60.5</td>
<td>67.1</td>
<td>55.9</td>
<td>70.8</td>
<td>55.3</td>
</tr>
<tr>
<td>BERT (re-trained)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our default</td>
<td>63.0 56.1</td>
<td>69.6 62.8</td>
<td>69.5</td>
<td>58.8</td>
<td>75.6</td>
<td>59.2</td>
</tr>
<tr>
<td>Our tuned</td>
<td>62.4 55.9</td>
<td>70.3 62.3</td>
<td>71.4</td>
<td>61.9</td>
<td>75.6</td>
<td>59.2</td>
</tr>
</tbody>
</table>

• Conclusions made earlier remain the same
• For more discussion on the re-training issue, see later slides
### Timing Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
<th>UNFAIR-ToS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM thresholding</td>
<td>28s</td>
<td>29s</td>
<td>1m 11s</td>
<td>4m 2s</td>
<td>28s</td>
<td>2s</td>
</tr>
<tr>
<td>cost-sensitive</td>
<td>59s</td>
<td>1m 0s</td>
<td>2m 11s</td>
<td>28m 8s</td>
<td>3m 26s</td>
<td>3s</td>
</tr>
<tr>
<td></td>
<td>1m 38s</td>
<td>1m 43s</td>
<td>3m 28s</td>
<td>50m 36s</td>
<td>4m 45s</td>
<td>4s</td>
</tr>
<tr>
<td><strong>BERT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our default</td>
<td>1h 2m</td>
<td>1h 2m</td>
<td>46m 52s</td>
<td>6h 38m</td>
<td>9h 15m</td>
<td>34m 46s</td>
</tr>
<tr>
<td>Our tuned</td>
<td>5h 17m</td>
<td>5h 33m</td>
<td>3h 28m</td>
<td>38h 17m</td>
<td>43h 58m</td>
<td>4h 13m</td>
</tr>
</tbody>
</table>

- Methods based on linear classifiers are much faster
- Moreover, they use CPU (Intel Xeon E5-2690) instead of GPU (4 NVIDIA V100)
# Model Size Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>ECtHR (A)</th>
<th>ECtHR (B)</th>
<th>SCOTUS</th>
<th>EUR-LEX</th>
<th>LEDGAR</th>
<th>UNFAIR-ToS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>924K</td>
<td>924K</td>
<td>2M</td>
<td>15M</td>
<td>2M</td>
<td>50K</td>
</tr>
<tr>
<td>BERT variants</td>
<td></td>
<td>110M ~ 149M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All three linear methods in LibMultiLabel have the same model size.
- For BERT variants, we borrow the calculation from Chalkidis et al. (2022).
- Linear SVM requires a much smaller model than BERT.
Lessons Learned

- Advanced models like BERT are very useful if they are properly used.
- However, sometimes a direct use leads to poor results.
- Further, techniques developed for long documents may not be consistently better than the baseline BERT (Park et al., 2022).
- For text classification, unless documents are very short, tf-idf features are informative. Thus, linear methods can serve as a simple but strong baseline.
Lessons Learned (Cont’d)

- Results from linear help to see if an advanced method has been properly applied
- But an issue I found is that many young students do not believe the usefulness of linear classifiers
Outline

1. Text classification and the project LibMultiLabel
2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models
3. Issues in designing a text classification package
   - Retraining after hyper-parameter search
4. Conclusions
Issues in designing a text classification package

Retraining after hyper-parameter search

Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
A common deep learning procedure:

- Split data to training/validation
- Conduct hyper-parameter search
- Return the model with the best validation performance
To Retrain or Not to Retrain?

- The final model does not use validation data for training.
- For other methods like SVM/LR, in general all data are trained under the best hyper-parameters (called the *retraining* procedure here).
- The reason is that more information is used.
- But for deep learning people may not do this step, why?
To Retrain or Not to Retrain?

- Some works such as Goodfellow et al. (2013); Srivastava et al. (2014); Goodfellow et al. (2016) have studied this issue.
- People may think that this is an old and solved issue.
- But in online forums, many practitioners still asked about this re-training issue.
- We did some studies and found that things are more complicated than we thought.
Early Stopping of Training

- Empirical risk only approximates the true risk, so accurate empirical risk minimization is not needed (Bottou and Bousquet, 2008).
- However, for convex problems like SVM/LR, we often do accurate minimization by check gradient for stopping.
- Reason: not too time-consuming and convenient.
- But for deep learning, early stopping is needed.
- Usually this is by checking the validation performance.
The Re-training Process

- Now all available data are used
- Thus optimization processes relying on validation data to terminate no longer work
### Possible Stopping Conditions

Properties of training the best model in hyper-parameter search may be used. For example,

- **Epochs/Max**: 18/50
- **Loss\(_{\text{train}}\)**: 0.044
- **Model**: \(f^*\)

#### Possible criteria

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Model initialization</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed-epochs</td>
<td>from scratch</td>
<td>re-train for 50 epochs</td>
</tr>
<tr>
<td>optimal-epochs</td>
<td>from scratch</td>
<td>re-train for 18 epochs</td>
</tr>
<tr>
<td>function-based</td>
<td>warm up by (f^*)</td>
<td>Loss(_{\text{val}}) matches 0.044</td>
</tr>
</tbody>
</table>
Sample Results

- Three NN models are checked (CNN, RNN, and BERT)
Analysis

- Re-training is beneficial for most problems
- But the best strategy seems to be model dependent
- Fixed-epochs more easily causes overfitting
- function-based: sometimes training loss dropped too quickly. Then overfitting occurs
- optimal-epochs: generally stable, but slightly worse than function-based on BERT
Detailed Analysis of an Example

Kim-CNN is applied on MIMIC-III

<table>
<thead>
<tr>
<th>Termination criterion</th>
<th>Epochs</th>
<th>Loss (train)</th>
<th>Loss (valid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no re-train</td>
<td>11.8±0.8</td>
<td>0.425±0.015</td>
<td>0.722±0.006</td>
</tr>
<tr>
<td>fixed-epochs</td>
<td>50.0±0.0</td>
<td>0.280±0.008</td>
<td>0.286±0.011</td>
</tr>
<tr>
<td>optimal-epochs</td>
<td>11.8±0.8</td>
<td>0.424±0.006</td>
<td>0.450±0.011</td>
</tr>
<tr>
<td>function-based</td>
<td>11.6±2.7</td>
<td>0.343±0.014</td>
<td>0.420±0.020</td>
</tr>
</tbody>
</table>

- fixed-epochs: model overfits the training data
- function-based: training loss becomes lower and overfitting seems to occur on the training subset
- optimal-epochs: the training and validation losses are similar to the one without re-training
Discussion

- For we package developers, should we by default do re-training after hyper-parameter search?
- If so, which setting should we provide?
- Can situations for other applications (e.g., in computer vision) different?
- Now we mainly have empirical evaluation. Can we develop some good theory for this re-training process?
Outline

1. Text classification and the project LibMultiLabel

2. Why inappropriate machine learning use is a concern
   - Story: inattention of parameter selection
   - Story: casual use of advanced models

3. Issues in designing a text classification package
   - Retraining after hyper-parameter search

4. Conclusions
In machine learning, we often seek for better algorithms or efficient systems.

However, to help people obtain satisfactory results, we need more than that.

How to guide users to effectively solve their problems is the ultimate goal we machine learning researchers should try to achieve.