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

Text Classification (Cont'd)

- 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

Outline

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

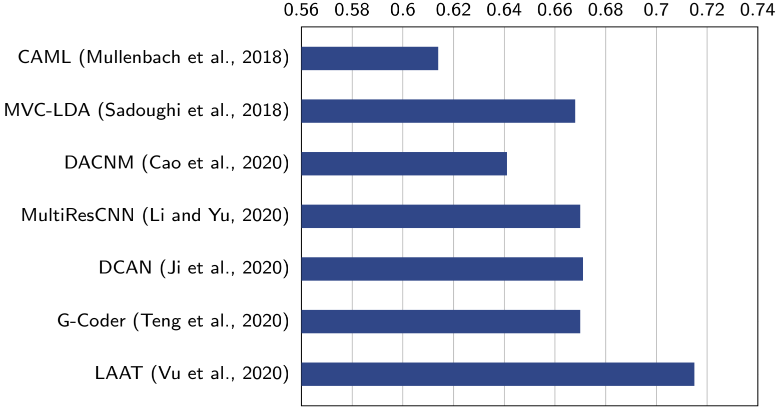
Multi-label Classification for Medical Code Prediction

- 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

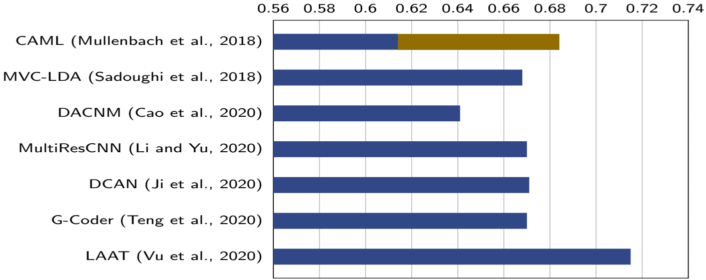


Did They Really Make Progress on MIMIC-III-50?

- 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 ...

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

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

Results in Chalkidis et al. (2022): Performance

Method	ECtHR (A)		ECtHR (B)		SCOTUS		EUR-LEX		LEDGAR		UNFAIR-ToS	
	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1
TF-IDF+SVM	64.5	51.7	74.6	65.1	78.2	69.5	71.3	51.4	87.2	82.4	95.4	78.8
BERT	71.2	63.6	79.7	73.4	68.3	58.3	71.4	57.2	87.6	81.8	95.6	81.3
RoBERTa	69.2	59.0	77.3	68.9	71.6	62.0	71.9	57.9	87.9	82.3	95.2	79.2
DeBERTa	70.0	60.8	78.8	71.0	71.1	62.7	72.1	57.4	88.2	83.1	95.5	80.3
Longformer	69.9	64.7	79.4	71.7	72.9	64.0	71.6	57.7	88.2	83.0	95.5	80.9
BigBird	70.0	62.9	78.8	70.9	72.8	62.0	71.5	56.8	87.8	82.6	95.7	81.3
Legal-BERT	70.0	64.0	80.4	74.7	76.4	66.5	72.1	57.4	88.2	83.0	96.0	83.0
CaseLaw-BERT	69.8	62.9	78.8	70.3	76.6	65.9	70.7	56.6	88.3	83.0	96.0	82.3

- μ-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

Results in Chalkidis et al. (2022): Time

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

- 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

Direct Run of Linear Classifiers

- 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

max_seq_length	learning_rate	dropout
[128, 512]	[2e-5, 3e-5, 5e-5]	[0.1, 0.2]

Performance Comparison

Method	ECtHR (A)		ECtHR (B)		SCOTUS		EUR-LEX		LEDGAR		UNFAIR-ToS	
	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1
Linear												
SVM	64.0	53.1	72.8	63.9	78.1	68.9	72.0	55.4	86.4	80.0	94.9	75.1
thresholding	68.6	64.9	76.1	68.7	78.9	71.5	74.7	62.7	86.2	79.9	95.1	79.9
cost-sensitive	67.4	60.5	75.5	67.3	78.3	71.5	73.4	60.5	86.2	80.1	95.3	77.9
BERT												
Our default	60.5	53.4	68.9	60.8	66.3	54.8	70.8	55.3	85.2	77.9	95.2	78.2
Our tuned	61.9	55.6	69.8	60.5	67.1	55.9	70.8	55.3	87.0	80.7	95.4	80.3
Chalkidis et al.	71.2	63.6	79.7	73.4	68.3	58.3	71.4	57.2	87.6	81.8	95.6	81.3

- 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

	ECtHR (A)	ECtHR (B)	SCOTUS	EUR-LEX	LEDGAR	UNFAIR-ToS
length	1,662.08	1,662.08	6,859.87	1,203.92	112.98	32.70

- But many may not know that BERT takes up to only 512 tokens

Performance Comparison (Cont'd)

- 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

Performance Comparison: Some Notes

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

Method	ECtHR (A)		ECtHR (B)		SCOTUS		EUR-LEX		LEDGAR		UNFAIR-ToS	
	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1	μ-F_1	m-F_1
Linear												
SVM	64.0	53.1	72.8	63.9	78.1	68.9	72.0	55.4	86.4	80.0	94.9	75.1
thresholding	68.6	64.9	76.1	68.7	78.9	71.5	74.7	62.7	86.2	79.9	95.1	79.9
cost-sensitive	67.4	60.5	75.5	67.3	78.3	71.5	73.4	60.5	86.2	80.1	95.3	77.9
BERT												
Our default	60.5	53.4	68.9	60.8	66.3	54.8	70.8	55.3	85.2	77.9	95.2	78.2
Our tuned	61.9	55.6	69.8	60.5	67.1	55.9	70.8	55.3	87.0	80.7	95.4	80.3
BERT (re-trained)												
Our default	63.0	56.1	69.6	62.8	69.5	58.8	75.6	59.2	85.3	78.4	94.0	65.4
Our tuned	62.4	55.9	70.3	62.3	71.4	61.9	75.6	59.2	87.2	81.5	95.2	79.8

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

Timing Comparison

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

- 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

Method	ECtHR (A)	ECtHR (B)	SCOTUS	EUR-LEX	LEDGAR	UNFAIR-ToS
Linear	924K	924K	2M	15M	2M	50K
BERT variants	$110M \sim 149M$					

- 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)

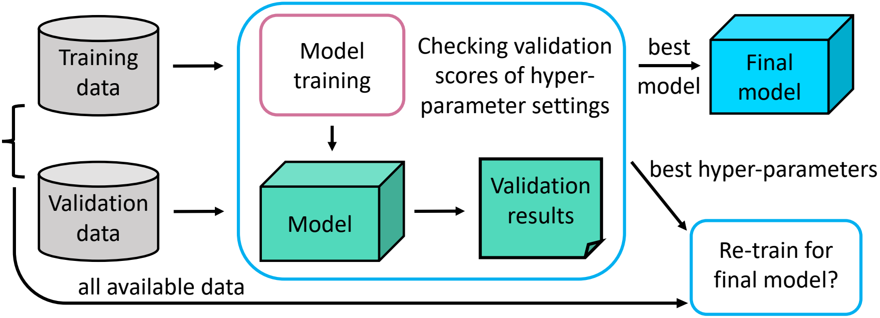
- Results from linear help to see if an advanced method has been properly applied
- But an issue I found is that many young students do not believe the usefulness of linear classifiers

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

Outline

An Issue After Hyper-parameter Search



- 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

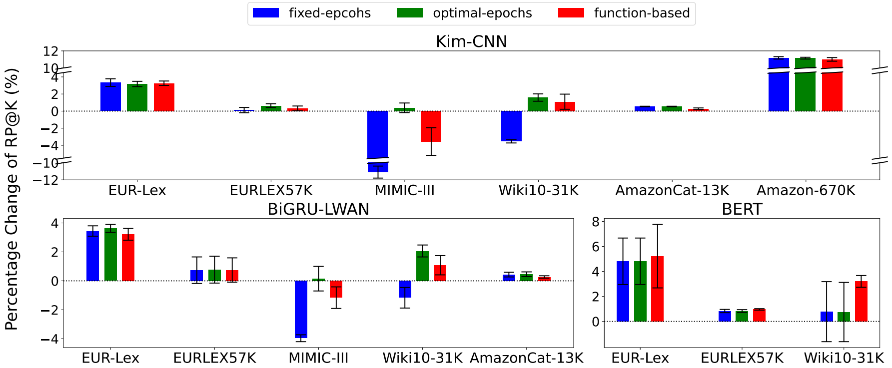
Properties of training the best model in hyper-parameter search may be used. For example,

Epochs/Max	18/50
$\text{Loss}_{\text{train}}$	0.044
Model	f^*

Possible criteria

Criterion	Model initialization	Termination
fixed-epochs	from scratch	re-train for 50 epochs
optimal-epochs	from scratch	re-train for 18 epochs
function-based	warm up by f^*	Loss_{val} matches 0.044

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

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

- 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

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