COMP0188 Coursework
Review on "Easy over Hard: A Case Study on Deep Learning"

Henning Heyen

December 22, 2022

1 Paper description

Even though Deep Learning shows some exciting results, they usually come with high computational costs and long training runtimes. The study *Easy over Hard: A Case Study on Deep Learning*" by Wei Fu and Tim Menzies [2] aims to show that there are simpler alternatives than neural networks that achieve similar results at much lower costs. For that, a case study on predicting the relatedness of Stackoverflow posts was carried out using a standard Support Vector Machine (SVM) model fine-tuned with a method called Differential Evolution (DE). The same task was studied by Xu [5] using a convolutional neural network (CNN) and used as a comparison.

1.1 Technical background

Xu's study predicted whether or not two posts, including the answers (*Knowledge Unit* or KU), are related to each other. Four different classification categories are defined, see Figure 1.

Class	Description
Duplicate	These two knowledge units are addressing the same question.
Direct link	One knowledge unit can help to answer the question in the other knowledge unit.
Indirect link	One knowledge provides similar information to solve the question in the other knowledge unit, but not a direct answer.
Isolated	These two knowledge units discuss unrelated questions.

Figure 1: labels for relatedness of Stackoverflow posts, borrowed from [2]

Xu uses a word embedding technique (*continuous skip-gram*) followed by an SVM algorithm as a baseline. The continuous skip-gram model is an "unsupervised word representation learning method based on neural networks" [2]. The word representations aim to predict surrounding words in a context window. The resulting vector representations serve as the input for SVM and the CNN. SVM's parameters were tuned using grid search, which, according to [2], is a method "heavily deprecated in the machine learning community". The proposed final model uses the same word embedding followed by a CNN. The training phase of the deep learning model from [5] "takes about 14 hours" [2] on a 2.5GHz Windows7 machine with 16 GB RAM. Xu's study was chosen for comparison because both the hardware and runtime of the CNN and the baseline model were reported.

1.2 Research approach

The study from [5] is repeated using the same baseline (Word Embedding + SVM) but the SVM is fined tuned using Differential Evolution (DE), which is a more sophisticated method to tune hyperparameters for classical ML algorithms developed by [4]. DE was shown to be more efficient and performs better than grid search [1,3] as it explores the parameter space in the direction of the most promising changes. Grid search, however, just loops over all combinations of the predefined parameters. During the evaluation phase, accuracy, precision, recall and F1 score are considered. DE is optimised for F1. Since Xu's study did not provide the code publicly, this study had to start by reimplementing the same baseline model (without DE) to minimise the risk of implementation bias when comparing the tuned SVM model with the deep learning model. To enable better comparison between both approaches, [2] uses the same training $(6,400)$ and test $(1,400)$ KU pairs as in [5]. For training the word embedding, $100,000$ posts that are tagged with "*java*" were used, just like in [5]. Unlike Xu's study, this paper split the $6,400$ training examples into new training data and tuning data used by DE to evaluate candidate parameters for SVM. For more robust results, 10-fold cross validation was applied on that split. The tuned SVM, was then evaluated on the test data. The whole workflow can be summarized in Figure 2.

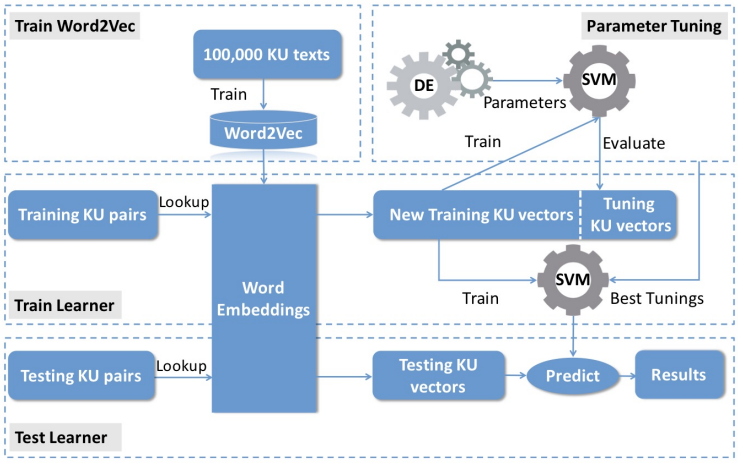


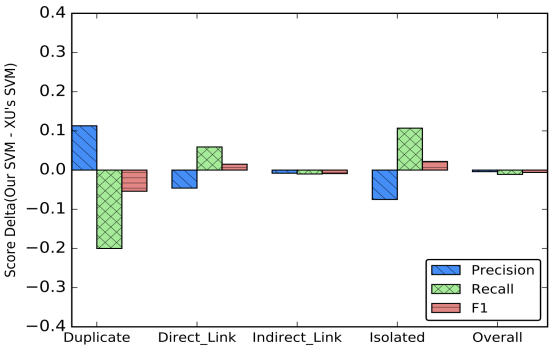
Figure 2: Workflow on tuned SVM, borrowed from [2]

1.3 Results

First, both baseline models (this study's SVM and Xu's SVM) were compared, see Figure 3 (all result plots are borrowed from [2].

As both baselines perform very similarly, [2] argues that any difference found in the comparison of Xu's CNN and this study's tuned SVM can be reduced to the parameter tuning of the SVM. As far as the DE-tuned SVM is concerned, it is not outperformed by the CNN from [5]. Figure 4 shows that in 8 out of 12 scores, the tuned SVM performed even better than the CNN. In the other four scores where the CNN performed better, the difference is less than 0.1. On average, the tuned SVM performed better by 0.238, 0.228 and 0.227 in precision, recall and F1-score for all four KU relatedness classes.

As for runtime analysis, according to [2] it "takes 10 minutes to run SVM with parameter tuning by DE on a similar environment". The CNN in [5] took 14 hours to train. That makes the tuned SVM 86 times faster than the Deep Learning method thanks to a simple parameter tuning method, i.e. differential evolution.

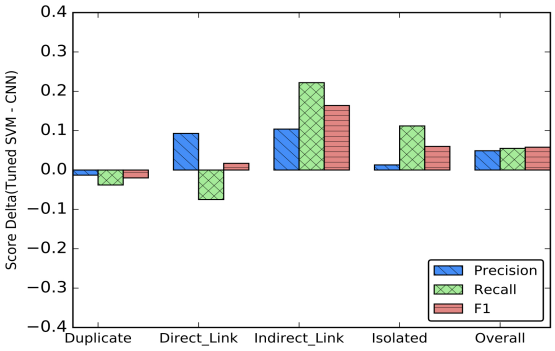


(a) $Fu'sSVM - Xu'sSVM$ in each category

Metrics	Methods	Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	Our SVM	**0.724**	0.514	0.779	0.601	0.655
	XU's SVM	0.611	**0.560**	**0.787**	**0.676**	**0.659**
Recall	Our SVM	0.525	**0.492**	0.970	**0.645**	0.658
	XU's SVM	**0.725**	0.433	**0.980**	0.538	**0.669**
F1-score	Our SVM	0.609	**0.503**	0.864	**0.622**	0.650
	XU's SVM	**0.663**	0.488	**0.873**	0.600	**0.656**
Accuracy	Our SVM	0.525	0.493	0.970	0.645	0.658
	XU's SVM	-	-	-	-	**0.669**

(b) Direct comparison of each metric

Figure 3: Comparison of both SVM baselines. "Our" refers to Fu's study [2]



(a) $TunedSVM - CNN$ in each category

Metrics	Methods	Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	XU's SVM	0.611	0.560	0.787	0.676	0.658
	XU's CNN	**0.898**	0.758	0.840	0.890	0.847
	Tuned SVM	0.885	**0.851**	**0.944**	**0.903**	**0.896**
Recall	XU's SVM	0.725	0.433	0.980	0.538	0.669
	XU's CNN	**0.898**	**0.903**	0.773	0.793	0.842
	Tuned SVM	0.860	0.828	**0.995**	**0.905**	**0.897**
F1-score	XU's SVM	0.663	0.488	0.873	0.600	0.656
	XU's CNN	**0.898**	0.824	0.805	0.849	0.841
	Tuned SVM	0.878	**0.841**	**0.969**	**0.909**	**0.899**

(b) Direct comparison of each metric

Figure 4: Comparison of Xu's CNN and the DE tuned SVM from [2]

2 Critical analysis and limitations of the work

This study clearly shows some exciting alternatives to a resource-intense deep learning model. As computation is shifted more and more to the cloud environment, which is highly monetized [2], the same approach could save time and financial costs.

One obvious question is: How well does this approach generalize to other applications? [2] acknowledges that just because SVM + DE performed as well as a CNN in this application "does not mean DE is always the superior method" [2] in every other application. Nonetheless, the paper gives incentives to first use a fine-tuned conventional machine learning algorithm as a baseline before deploying CPU and GPU-intensive deep learning models. The study also lists is a wide range of alternatives to differential evolution (e.g. simulated annealing, genetic algorithms, tabu search, scatter search) which are more effective methods than basic grid search [1].

Even though [2] puts effort into making a fair comparison between the CNN and the tuned SVM, there are some flaws in the internal validity of that approach. For example, the CNN was run on a similar but different hardware environment than the tuned SVM, which could introduce biases, see Figure 5.

Furthermore, both SVM baselines perform similarly, but for the *Duplicate* category, there are significant differences in the reimplementation as depicted in Figure 3a. The difference in recall and precision goes as far as -0.2 and 0.1, respectively. Therefore, concluding that no implementation biases are involved when comparing the tuned version with the CNN is not completely coherent. The

Methods	OS	CPU	RAM
Tuning SVM	MacOS 10.12	Intel Core i5 2.7 GHz	8 GB
CNN	Windows 7	Intel Core i7 2.5 GHz	16 GB

Figure 5: Hardware environment comparison

most straightforward mitigation for this problem, of course, would be if [5] makes its implementation open source (just like this study[1]).

Lastly, [2] fails to mention some technical details. For example, the word embedding technique, continuous skip-gram, is also "based on neural networks" [2], but there is no separate runtime analysis for the word embedding. Also, it would be helpful if [2] would comment on the CNN architecture of [5] to get insights into the complexity and the resulting computational concerns.

3 Potential future work and implementation in real-world applications

As the amount of Stackoverflow posts keeps growing, real-world Machine Learning models might require much more training data to capture their relatedness. As for this paper, the training size was quite low (6,400 knowledge units). It would be interesting to see how the tuned SVM would perform for larger training sets, especially in comparison with deep learning models such as the CNN in [5]. While deep learning models tend to improve as the training dataset increases, more conventional models, such as SVM, are known to struggle. The runtime improvements might come along with less performance. More experimentation in that regard will benefit the discussion.

More generally, it would be interesting to see more case studies where fine-tuned standard ML models are compared with state-of-the-art deep learning models to increase the external validity of this paper's claim. At least for text data, this study gives hope that less resource-intense architectures can achieve as good performance. [2] names a list of deep learning applications where a comparison with a similar approach might be worth studying (e.g. bug localization, clone code detection, malware detection, API recommendation etc.).

Lastly, developers would be motivated to use more sophisticated tuning algorithms like differential evolution if they become part of a standard toolkit, just like grid search is part of the scikit-learn library. Differential evolution, for example, is currently "often written from scratch" [3], which can be tedious.

[1] https://github.com/WeiFoo/EasyOverHard

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[2] Wei Fu and Tim Menzies. Easy over hard: A case study on deep learning. In *Proceedings of the 2017 11th joint meeting on foundations of software engineering*, pages 49–60, 2017.

[3] Wei Fu, Vivek Nair, and Tim Menzies. Why is differential evolution better than grid search for tuning defect predictors? *arXiv preprint arXiv:1609.02613*, 2016.

[4] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[5] Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 51–62. IEEE, 2016.