When data permutations are pathological: the case of neural natural language inference

Natalie Schluter and Daniel Varab
Department of Computer Science
IT University
Copenhagen, Denmark
{natschluter, djam}@itu.dk

Abstract

Consider two competitive machine learning models, one of which was considered state-of-the art, and the other a competitive baseline. Suppose that by just permuting the examples of the training set, say by reversing the original order, by shuffling, or by mini-batching, you could report substantially better/worst performance for the system of your choice, by multiple percentage points. In this paper, we illustrate this scenario for a trending NLP task: Natural Language Inference (NLI). We show that for the two central NLI corpora today, the learning process of neural systems is far too sensitive to permutations of the data. In doing so we reopen the question of how to judge a good neural architecture for NLI, given the available dataset and perhaps, further, the soundness of the NLI task itself in its current state.

1 Introduction

There is increased interest today in the detection of information quality: whether a statement is true or false, or equivalently, whether one statement (the premise) is entailed by, contradicts or has no relation to another statement (the hypothesis). This is the Natural Language Inference task. The timely development of the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and more recently the Multi-Genre NLI (MULTI-NLI) corpus (Williams et al., 2017) has lead to a steady increase in contributions to research on NLI. Recent research, however, indicates these central datasets to be trivially annotated to a large degree (Gururangan et al., 2018). In this paper, we give evidence of an unrelated problem.

Deep neural network approaches provide the state-of-the-art today for the tasks. We show a pathological sensitivity of these systems to permutations of the training set. This calls into question the soundness of the task in its current state and corresponding development and benchmarking of NLI neural systems.

Given the approximate iterative optimisation methods required to induce models, a common practice for statistical learning is to first randomly shuffle the training set. This serves to offset unwanted bias due to accidental ordering of the examples (for example, with respect to time or class). Probably because of this, there is very little literature or understanding of the effect of order among the training examples–strict ordering of examples has simply been known to be undesirable. However, as neural network approaches increasingly dominate in performance across many NLP tasks, the notion of random shuffling has become overshadowed by that of computational efficiency. This seems to lead back to experiment parameters from Sutskever et al. (2014)'s work, which demonstrates that grouping examples of similar sentence length into batches results in a halved training time. But what is the effect on accuracy? Indeed, a curiosity of neural network models induced over the NLI datasets is the ease in acquiring rather instable results as a result of simple example order permutations of the training set.

In this paper, we present an investigation into these questions. In particular, we consider two simple but competitive neural network topologies in order to investigate the effect of training example order from these datasets on performance. One of these achieves close to state-of-art results over SNLI and state-of-the-art for MULTI-NLI (for simple systems, Cf. Section 4.1), while the other is a simpler variant of the first, characterised by fewer parameters. Out of the standard deep learning, standard machine learning and other plausi-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4935–4939
Brussels, Belgium, October 31 - November 4, 2018.

ble orderings of the dataset, we show that only the original ordering of the training examples leads to state-of-the-art model induction. We also show that the gap in performance between the two neural architectures described here generally drops over all other permutations.

2 NLI task and datasets

The NLI task input is two sentences, $\mathbf{a} = (a_1, \ldots, a_{l_a})$ and $\mathbf{b} = (b_1, \ldots, b_{l_b})$ of lengths l_a and l_b respectively. Each a_i (resp. b_j) for $i \in [l_a]$ (resp. $j \in [l_b]$) corresponds to the word embedding with dimension d for the ith (resp. jth) word. The task dataset consists of labeled pairs of sentences, $\{\mathbf{a^{(n)}}, \mathbf{b^{(n)}}, \mathbf{y^{(n)}}\}_{n=1}^{N}$, where $y^{(n)} \in \{\text{entailment}, \text{contradiction}, \text{neutral}\}$ is the class.

We use two central datasets for our study here: SNLI and MULTI-NLI (Bowman et al., 2015; Williams et al., 2017), containing over 570K and 433K sentence pairs respectively.

3 Our neural network architectures

For the research presented in this paper, we choose two related and relatively simple neural network architectures corresponding roughly to Parikh et al. (2016) and a simplified version of Chen et al. (2017) consisting of five components.[1] Note that we also downloaded Chen et al. (2017)'s code[2] and ran it over the datasets; since it did not perform with the same accuracy reported in the paper for our run (87.77% instead of the reported 88%), and since it took several hours longer than our implementation to run (+7 hours longer), we concentrated on our own implementation. We also attempted to run Gong et al. (2017)'s system[3]; for our runs, the system halted without completion after several hours. Moreover, there are reports on the difficulty in getting the architecture to achieve the accuracy score of 88% reported in the original paper, Mirakyan et al. (2018) reporting that their re-implementation could only achieve 86.38%. This is significantly worst-performing than our best system, which we present now.

(1) Pre-projection. To compensate approximately for not updating the original embeddings during learning, we first carry out a preliminary projection of the embeddings, to the same dimensions using a feed-forward network.

(2) Embedding projection. We further project embeddings via either a simple feed-forward (FF) with a ReLU activation function or a bidirectional LSTM (BiLSTM) layer. The result is then sent to the attention component. This corresponds precisely therefore to Parikh et al. (2016)'s computationally efficient approximation of the vector product before soft-alignment.

(3) Attention. The attention mechanism, first introduced by Bahdanau et al. (2015), is based on a matrix of all-pairs scores between the elements of two sequences $\mathbf{a_i}$ and $\mathbf{b_j}$:[4]

$$e_{ij} := F'(\mathbf{a_i}, \mathbf{b_j}) := F(\mathbf{a_i})^T F(\mathbf{b_j})$$

We follow (Parikh et al., 2016) and later attention-based models for NLI, by representing the importance of $\mathbf{a_i}$ with respect to $\mathbf{b}$ as the normalised sum

$$\beta_i := \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} b_j.$$

The result is concatenated with $\mathbf{a_i}$, to create $[\mathbf{a_i}, \beta_i]$ which is then projected down to original embedding dimension. The same is done for $\mathbf{b_j}$ with respect to $\mathbf{a}$.

(4) Aggregation. Following the attention mechanism we aggregate over words for a sentence representation. Either we sum over the attended word vectors Parikh et al. (2016) or use the final state of a single LSTM layer (Chen et al., 2017)[5].

(5) Prediction. Finally we feed a vector concatenation of both sentence vectors as input to a component consisting of three feed-forward layers with dropout and regularisation, followed by a linear softmax layer for prediction.

Instantiated architectures. The two topologies we adopt for this study consist of the above components with embedding projection and aggregation as follows:

- FF/SUM corresponding to the embedding projection instantiated with a feed-forward network and the aggregation carried out through vector summation, and

[1] We make the systems publicly available https://github.com/natschluter/nli-data-permutations.

[2] https://github.com/lukecq1231/nli

[3] https://github.com/YichenGong/Densely-Interactive-Inference-Network

[4] Note that we also experimented with $e_{ij} := F'(\mathbf{a_i}, \mathbf{b_j})$ where F' was a component-wise multiplication and F' a feed-forward network with ReLU activations. However, we observed no benefits to doing so performance-wise, and substantial loss in computational efficiency.

[5] For Chen et al. (2017), aggregation consists of several layers of BiLSTMS as opposed to our single LSTM layer.

- Bi/LSTM corresponding to the embedding projection instantiated with a BiLSTM and the aggregation carried out via an LSTM.

Other hyperparameters. We use 300 dimensional GloVe embeddings trained on the Common Crawl 840B tokens dataset (Pennington et al., 2014), which remain fixed during training. Out of vocabulary (OOV) words are represented as zero vectors.[6] We use a 0.2 dropout rate and L2 regularisation, applied in all feed-forward layers. We optimise the network using categorical cross-entropy loss and employ the RMSprop optimizer with ρ set to 0.9, a 0.001 learning rate, with a batch size of 512 and use early stopping over the development set after no improvement in accuracy after 4 epochs.

State-of-the-art for simple systems. The models are **simple** in that no information above word embeddings is taken as input, for example, no POS-tags or syntactic relations (See Section 4.1). Our Bi/LSTM system also currently sets the state-of-the-art for simple systems on the MULTI-NLI dataset (See Section 4.1).

Proj- Agg	split	SNLI accuracy	MULTI-NLI accuracy
FF/ SUM	test	85.72	71.58
	test-mis	-	70.63
	dev	85.64	72.29
	dev-mis	-	72.80
Bi/ LSTM	test	**87.12** (+1.4)	**75.58** (+4.0)
	test-mis	-	**74.33**
		-	(+3.7)
	dev	**87.53**	**76.43**
		(+1.89)	(+4.14)
	dev-mis	-	**76.04**
		-	(+3.24)

Table 1: Performance in accuracy of networks, by projection (Proj-) and aggregation (Agg) components.

4 Related work

Previous work is related either in terms of the neural architectures for NLI (Section 4.1), or in terms of work on training data permutations in learning (Section 4.2).

4.1 State-of-the-art NLI

There are different types of neural network systems in the literature with respect to the simplicity of input data required for modeling and interdependence of the internal modules. In this work, we only consider simple system approaches that use only word embeddings (no character representations, POS-tags, word-position, syntactic tree, external resources, etc.) and consist only of interdependent modules (not ensembles). We make no claims regarding linguistic or ensemble-complex systems, but make the straightforward hypothesis that the conclusions presented here can be extended to cover more complex frameworks as well, especially given that our systems are strongly competitive or even better performing than two highly complex state-of-the-art neural systems (Cf. Section 3).

For simple systems, the state-of-the-art is currently set by Sha et al. (2016) at 87.5%, on SNLI. They use a standard BiLSTM to read the premise, and propose a Bi-rLSTM to read the hypothesis. Their proposed rLSTM–"re-read" LSTM unit–takes the attention vector of one sentence as an inner state while reading the other sentence. The output of the standard BiLSTM is also taken as the general input of the bidirectional rLSTM. The currently published next best performing simple system, Parikh et al. (2016) at 86.3% accuracy, introduced use of the attention mechanism for the NLI task, the way it is generally being used today.[7]

4.2 Example permutation in learning

Morishita et al. (2017) explored the effect of minibatching on the learning of Neural Machine Translation models, carrying out their experiments on two datasets (two language-pairs). In particular, they studied the strategies of (1) sorting by length of the source sentence, (2) target sentence, or (3) both, among other things. They empirically compare their efficiency on two translation tasks and find that some strategies in wide use are not necessarily optimal for accuracy and convergence-wise. In contrast to the work described here however, one of the sorting strategies produced best results, though no comparison was made with the original ordering of examples. By contrast, we show that it is by making non-canonical (semi-non-random) orderings of the data that best results are achieved in NLI, for the two available datasets.

[6] We experimented with alternative OOV representations, such as the the mean vector of most frequent words and random vectors, with insignificant effects.

[7] Parikh et al. (2016) also have a result with intra-sentential attention that performs with 86.8% accuracy. However this attention model is dependent on word positional information.

permutation	SNLI FF/SUM	SNLI Bi/LSTM	SNLI diff	MULTI-NLI FF/SUM	MULTI-NLI Bi/LSTM	MULTI-NLI diff
orig	85.64	87.53	1.89	72.29	76.43	4.14
orig-r	82.48	83.92	1.43	66.71	69.71	3.0
shuffle-once (5 runs)	82.88 ($\sigma = 0.19$)	84.15 ($\sigma = 0.18$)	1.27	72.39 ($\sigma = 0.08$)	75.86 ($\sigma = 0.18$)	3.47
shuffle-epoch	83.09	84.69	1.60	72.15	75.65	3.50
conf	83.35	83.96	0.61	67.12	69.29	2.17
conf-r	81.4	83.95	2.55	66.93	69.25	2.32
prem	83.0	83.74	0.74	67.56	70.11	2.55
prem-r	82.68	83.49	0.81	66.6	69.7	3.1
hypo	82.61	83.82	1.22	67.71	70.03	2.31
hypo-r	82.42	83.68	1.26	66.97	69.77	2.8
lengths	82.74	83.69	0.96	67.11	70.04	2.92
lengths-r	82.65	83.55	0.9	66.93	69.93	3.01

Table 2: Performance of the FF/SUM and Bi/LSTM neural networks for the training example permutations, evaluated over the development set, along with the difference (diff) in performance between the two architectures.

5 Experiments

Data permutations. We consider the following original and sorted orderings of the training set examples to learn our models.

orig: The original order in which the dataset is currently distributed.

prem: Sorting by increasing premise length.

hypo: Sorting by increasing hypothesis length.

lengths: Sorting by increasing premise + hypothesis length.

conf: Sorting by increasing score, where the scores for each example are generating in training Bi/LSTM over orig.

shuffle-once: Randomly shuffle once before training, averaged over 5 runs.

shuffle-epoch: Randomly shuffle on each epoch during training, for a single run.

In addition to each of these, we consider the reversal of the order (indicated by the suffix **-r**.)

In order to not exhaust the test set, we generate the results over data permutations on the development set. These are given in Table 2.

Discussion. For both datasets, we observe that all other training example permutations result in a substantial drop in performance, by approximately 3-4% on SNLI and 1-6% on MULTI-NLI. Even a simple reversal of the original order leads to a substantial drop in performance. Shuffling consistently the data provides the strongest alternative training conditions to the original ordering.

Moreover, the difference in performance of the two separate architectures is generally much lower on all other permutations of the training data, calling into question the significance of the more complex components. These observations apply to both randomly shuffling (as advised in statistical learning practise) and ordering the data by length (as advised in deep learning for NLP practise).

For an analysis of the sorting permutations, we looked into whether hypothesis sentence lengths differed by class to such an extent that the dataset became sorted by class label. We cannot include the results here due to space constraints. However, we observed that ordering by class results in small (and quite interesting) drops in performance, so long as the original order is preserved. If we first randomly shuffle the examples before ordering them by class, similar drops in performance result to those in Table 2.

6 Conclusions

We have shown that models induced over the SNLI and MULTI-NLI datasets are greatly affected by the permutation of the training data instances at hand: recommended statistical learning or deep learning engineering strategies for ordering the training examples result in significantly and even substantially worse performance over these datasets. Our models are simple (no information over word embedding representations of sentences and no ensembles), but strongly competitive with (or better performing than) both SOTA (re-)implementations of much more complex neural systems. We make the straightforward hypothesise that these observations will extend to more complex models; we leave this to be verified by future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2015. A large annotated corpus for learning natural language inference .

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. ACL, Vancouver, BC, Canada.

Yichen Gong, Heng Luo, and Jian Zhange. 2017. Natural language inference over interaction space. Technical report. ArXiv:1709.04348 [cs.CL].

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, New Orleans, Louisiana, USA.

Martin Mirakyan, Karen Hambardzumyan, and Hrant Khachatrian. 2018. Natural Language Inference over Interaction Space: ICLR 2018 Reproducibility Report. *ArXiv e-prints* .

Makoto Morishita, Yusuke Oda, Graham Neubig, Koichiro Yoshino, Katsuhito Sudoh, and Satoshi Nakamura. 2017. An empirical study of mini-batch creation strategies for neural machine translation. In *The First Workshop on Neural Machine Translation (NMT)*. Vancouver, Canada.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc of EMNLP*. Austin, Texas, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2870–2879.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*.

A. Williams, N. Nangia, and S. R. Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *ArXiv e-prints* .