Are Deep Sequence Classifiers Good at Non-Trivial Generalization?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advances in deep learning models for sequence classification have greatly improved their classification accuracy, specially when large training sets are available. However, several works have suggested that under some settings the predictions made by these models are poorly calibrated. In this work we study binary sequence classification problems and we look at model calibration from a different perspective by asking the question: Are deep learning models capable of learning the underlying target class distribution? We focus on sparse sequence classification, that is problems in which the target class is rare and compare three deep learning sequence classification models. We develop an evaluation that measures how well a classifier is learning the target class distribution. In addition, our evaluation disentangles good performance achieved by mere compression of the training sequences versus performance achieved by proper model generalization. Our results suggest that in this binary setting the deep-learning models are indeed able to learn the underlying class distribution in a non-trivial manner, i.e. by proper generalization beyond data compression.

1 Introduction

Recent advances in deep learning models for sequence classification have greatly improved their classification accuracy, specially when large training sets are available. However, several works have suggested that under some settings the predictions made by these models are poorly calibrated: a model might provide in general correct predictions but give bad estimates of the confidence scores $\Pr(y|x)$. In other words, the estimates do not correspond to frequencies in the true distribution.

Having well calibrated estimates of $\Pr(y|x)$ is crucial to many machine learning applications. For example, this is specially critical in systems in which an end user needs to make a decision based on the prediction of the model. In this case having a good confidence estimate is as important as having a good prediction. Pipeline models are another setting in which proper modeling of the uncertainty of the prediction is necessary, because the output of one predictor is the input to another model. In this case, good confidence predictions are essential for modeling uncertainty along the pipeline and avoiding error propagation.

In this work we study binary sequence classification with a focus on sparse sequence classification tasks, by which we mean binary classification problems that have the property that only a very small fraction of the sequences in the domain belong to the target class. These types of problems naturally arise in several applications. For example, in Natural Language Processing, think of all possible sequences of words that constitute a valid English sentence, it is clear that out of all the possible word combinations only a very small fraction are valid words (i.e. should have non-zero probability). In Computational Biology, detecting relevant (distinguishing) segments in DNA sequences is known to

Submitted to Workshop on Robustness in Sequence Modeling at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

be a very challenging task, and it is postulated that only about 1–3% of sequence segments have any biological significance [Eskin, 2002].

While traditional evaluations on model calibration have focused on the quality of the conditional prediction $\Pr_M(y|x)$ in this paper we look at model evaluation from a different perspective by asking the question: Are deep learning models capable of learning the underlying target distribution? The main idea is quite simple, if we have access to large quantities of unlabeled data that accurately represent the input distribution we can estimate the joint distribution $\Pr_M(y, x)$ implicitly induced by the conditional model and then we can compare it to the true distribution.

More specifically, we propose an evaluation that compares the moments of the two distributions, that is the expectations assigned to sub-sequences of various lengths. Our proposed evaluation distinguishes between good performances achieved by mere compression of the training sequences and good performances achieved by proper model generalization.

We conduct experiments using three deep-learning models: an LSTM, a Transformer and a WFA (i.e. Weighted Finite Automata, a form of RNN with linear activation). Our results suggest that in the binary classification setting the deep-learning models are indeed able to learn the underlying class distribution in a non-trivial manner, i.e. by proper generalization beyond data compression. These results seem to be consistent with previous studies showing that deep-learning models provide calibrated predictions for binary classification problems [Niculescu-Mizil and Caruana, 2005].

Our main contributions are:

- We present a novel evaluation framework for sequence prediction models. By exploiting unlabeled data, we propose to evaluate the model with respect to the implicitly induced joint distribution.
- Our evaluation distinguishes performance over components of the distribution seen in training data as well as unseen components, thus explicitly differentiating compression from generalization.
- Our experiments on a sparse sequence classification task comparing three deep-learning architectures show that the models are able to induce good distributions in a non-trivial manner.

The paper is organized as follows: Section 2 discusses related work, Section 3 describes the proposed evaluation approach including the learning setting, the models and the evaluation metrics, Section 4 presents the experiments and discusses the results and finally Section 5 concludes the paper.

2 Related Work

In the recent literature several works have addressed the problem of model calibration. Some of these studies have shown that for some deep learning architectures, the predictions produced by the model are not well calibrated [Guo et al., 2017] [Kumar and Sarawagi, 2019]. This is specially true for non pre-trained transformers [Desai and Durrett, 2020]. In contrast, for the case of binary classification some previous work has suggested that they are indeed well calibrated [Niculescu-Mizil and Caruana, 2005]. For this binary setting there have been several proposals for model Recalibration, including isotonic regression [Niculescu-Mizil and Caruana, 2005] and Platt scaling [Platt, 1999]

Most of the prior art on model calibration has focused on evaluating the quality of the conditional class predictions $\Pr(y|x)$ but to the best our knowledge we are the first ones to study calibration by looking at the joint distribution $\Pr(x, y)$ induced by the learned classifier.

Calibration aside, several works have compared the classification performance of different deep-learning architectures. CNNs and RNNs [Józefowicz et al., 2016] [Yin et al., 2017] have been shown to give similar performance in multiple evaluation scenarios. Transformers on the other hand have been shown to outperform RNNs [Karita et al., 2019] [Lakew et al., 2018] especially in the context of NLP applications that exploit large pre-trained models such as BERT [Devlin et al., 2019]. More recently, transformers and CNNs have been also studied in the context of computer vision [Kolesnikov et al., 2021] [Pinto et al., 2021] [Bai et al., 2021]. Finally, there have been some experimental studies comparing WFAs to CNNs [Quattoni and Carreras, 2020] or RNNs [Quattoni and Carreras, 2019] showing that these simpler models can also be competitive in certain settings.

Finally, an orthogonal related problem is that of developing deep learning models for density estimation. These include Transformers [Fakoor et al., 2020], Autoregressive Networks [Uria et al., 2016] [Oliva et al., 2018] and Flow Models [Durkan et al., 2019] [De Cao et al., 2020]. We believe that these works are orthogonal to the goal of this paper because our focus is not to learn a good distribution but rather to propose an evaluation methodology to test the quality of the distributions implicitly induced by a given sequence classifier.

3 Evaluating Deep-Learning Model as Moment Predictors

This section describes the overall evaluation approach, we begin by detailing the learning setting and the models that we will compare, we then describe each of the proposed evaluation metrics.

3.1 Learning Setting

Our goal is to evaluate models defined over sequences of discrete symbols. More precisely we consider an alphabet Σ and the set of all possible sequences $\Sigma^\star$. In general, we can think of probabilistic binary sequence classifiers as functions from $\Sigma^\star \to [0, 1]$.

In our setting we will assume that we have access to a large set of sequences $U = \{x^{(1)}, \ldots, x^{(u)}\}$ sampled according to the underlying distribution over $\Sigma^\star$. We can think of U as a large set of unlabeled sequences that represent the domain of the sequence classification task. The target class Y that we wish to learn is a subset of sequences in U, that is $Y \subset U$. We are particularly interested in cases where the target class is rare, that is cases in which $|Y|$ is significantly smaller than $|U|$.

We will create a labeled training set $T = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ of size m by sampling sequences from U and labeling $y = 1$ as positive instances if they appear in Y, and otherwise as negative instances $y = 0$. Section 4 contains further details about how we create the task data from an existing dataset, for various training sizes m.

Given a training set we will train a sequence classifier M that defines a distribution $\Pr_M(y|x)$. The next section describes the models that were used in our experiments.

3.2 Models

In this work, we consider three sequence classification models: an **RNN**, a **Transformer** and a **WFA**.

As recurrent neural network (RNN) we employ a multi-layer LSTM [Hochreiter and Schmidhuber, 1997] with a binary classification feed-forward layer on top. For the Transformer we select the BERT [Devlin et al., 2019] architecture, and in particular the *bert-base* model, with a binary classification output layer on top. BERT is pre-trained on a huge amount of text data and, as we are dealing with proteins, we don't use the pre-trained weights and we expand the embeddings with new randomly initialized ones to deal with the protein dataset vocabulary. Given a sentence x and the set of class labels $L = \{0, 1\}$, when taking in input x both models output the conditional probability $\Pr(l|x)$ where $l \in L$.

In addition we will evaluate a WFA (Weighted Finite Automata), which in essence is an RNN with linear activation functions. For more details about the relations between WFAs and RNNs see [Rabusseau et al., 2019]. As a WFA classifier architecture we used the ensemble proposed in [Quattoni and Carreras, 2020], which combines independently-trained class-conditional WFAs to make its predictions.

3.3 Evaluation Metrics

The sequence model M that we wish to evaluate defines a conditional distribution $\Pr_M(y|x)$. Generally, we would like to compute the true error of M by evaluating the model on the true distribution $\Pr(x, y)$. In this section, we will use the set U as a proxy for $\Pr(x)$, and specifically we will estimate

the *moments* of M on the joint distribution $\Pr_M(x, y)$ that is implicitly induced by the model over the target class.[1]

More precisely, consider the set:

$$Z_n = \{z_n \mid z_n \in \Sigma^n, \exists x \in U \textit{ such that } z_n \in x\} \tag{1}$$

where z_n is a sub-sequence of size n, $z_n \in x$ indicates that z_n is a sub-sequence of a domain sequence x, and therefore Z_n is the set of all sub-sequences of size n observed in U. We use these sub-sequences as the support set to compute moments of a model, and we define the *moment* function $E_M : z_n \times U \to \mathbb{R}$ as:

$$E_M(z_n, U) = \frac{1}{|U|} \sum_{x \in U} \Pr_M(y|x) \, \#[z_n \in x] \tag{2}$$

where z_n is a sub-sequence of length n and $\#[z_n \in x]$ is a function that counts the number of times that z_n appears in sequence x. From now on, we will assume a fixed U and drop it from the expressions, and we will refer to the sub-sequences z_n and their values $E_M(z_n)$ as the *moments* of M.

Since we assume that U is a good representation of the the domain, $E_M(z_n)$ can be regarded as an estimate of the expected number of times that we should observe z_n in a sample from $\Pr_M(x, y)$.

Given sets U and Y, the *moments* from the true joint distribution $\Pr_Y(x, y)$ can be computed similarly as:

$$E_Y(z_n) = \frac{1}{|U|} \sum_{x \in Y} \#[z_n \in x] \tag{3}$$

Putting it all together, if we have a domain represented by a set of unlabeled sequences U, a subset of target sequences Y and a model $\Pr_M(y|x)$, we can compute the model's *moment* function $E_M(z_n)$ and compare it to the true *moments* $E_Y(x_n)$.

Notice that for a model that learns the target class perfectly (i.e. a model that gives accurate predictions and that it is perfectly calibrated) we will have:

$$E_M(z_n) = E_Y(z_n) \tag{4}$$

for all *moments* in N.

Now that we have the necessary functions, we propose to evaluate a model M by comparing E_M and E_Y. We still need to define how exactly the two *moment* functions are going to be compared. We propose the following metrics:

- **MSPC**: This metric measures the Spearman rank correlation between gold and model *moments* of a fixed length. The Spearman correlation coefficient measures the strength and direction of association between ranked variables. In the context of our evaluation a high MSPC means that the model sorts the *moments* of the distribution in a way that is similar to the gold ordering. The MSPC is computed as:

$$\frac{\mathrm{cov}(\mathrm{rank}[E_M(z_n)], \mathrm{rank}[E_Y(z_n)])}{\sigma(\mathrm{rank}[E_M(z_n)]) \cdot \sigma(\mathrm{rank}[E_Y(z_n)])} \tag{5}$$

  where $\mathrm{rank}[E(z_n)]$ are the raw function scores for all $z_n \in Z_n$ converted to ranks.

  In essence, this metric measures the agreement on partial orderings induced by the model and gold *moment* functions. That is, given any two pairs of *moments* of a fixed length z_n and z'_n whenever $E_Y(z_n) > E_Y(z'_n)$ we want $E_M(z_n) > E_M(z'_n)$.

- **MSPCP**: While MSPC is a useful metric when evaluating the class distribution induced by a model, we need to keep in mind that for longer *moments* $E_Y(z_n)$ will be sparse. In other words, the function will be 0 for most inputs since we are interested in evaluating learning scenarios in which the target class is rare. So in this context it can also be interesting to

[1] We highlight that the set U is assumed to be sampled from the true $\Pr(x)$ and will contain repetitions of each distinct x proportional to its likelihood. And so does Y. This detail is relevant because we can think of U as an empirical estimate of $\Pr(x)$, which is necessary for the definitions of moments in this section.

compute the Spearman rank correlation between gold non-zero *moments* and the model prediction for those *moments*, this will give an idea of how well the model would sort the *moments* of the target distribution if it knew the true support of the target *moments* function.

- **MR**: For longer *moments* the MSPC metric might be overly harsh, because most of the target *moments* will be 0 and among the ones that are not 0 there might be many ties. A complementary alternative metric in this case is the mean-rank given to the target non-zero *moments*:

$$1 - \sum_{z_n \,:\, E_Y(z_n)>0} \frac{\text{rank}[E_M(z_n)]}{|Z_n|} \tag{6}$$

  This metric captures how well a model performs in the task of making the binary prediction of which *moments* should be non-zero for the target class.

We believe that when evaluating a model it is important to make a distinction between the ability of a model to compress data versus its ability to generalize to unseen data. A natural way to achieve this goal in the context of our proposed *moments* evaluation is to make a distinction between *moments* that have been seen in the training set, versus *moments* that have not been seen in the training set. That is for any training set T and *moment* length n we can define a subset $Z_n^T \subseteq Z_n$:

$$Z_n^T = \{z_n \in Z_n : z_n \; appears \; in \; T\} \tag{7}$$

Since all of our metrics are defined over a set of *moments*, given any training set T we can define their analogous *unseen* evaluation by computing the metric over $Z_n \setminus Z_n^T$. We call these new metrics: **MSPC-U**, **MSPCP-U** and **MR-U**.

In order to evaluate the generalization ability of a model we will also consider a baseline model that memorizes the training data and makes predictions with the observed training *moments*. More precisely given a training set T the baseline will compute the moments function as:

$$E_{B_T}(z_n) = \frac{1}{|U|} \sum_{(x,y)\in T : y=1} \#[z_n \in x] \tag{8}$$

4 Experiments

This section describes all the details of our experiments, we begin by describing the data used, we then give details about how the models were optimized and we finish with a discussion of the results.

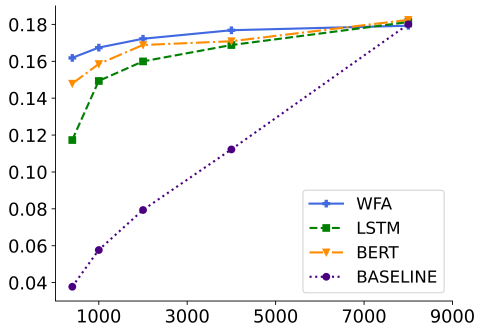
4.1 Data

Table 1: Dataset statistics.

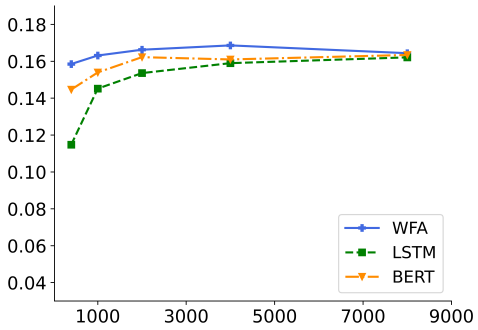
	# of sequences	**% wrt** $|\cup_n Z_n|$
$|\cup_n Z_n|$	1.8×10^6	-
$|Z_1^Y|$	2×10^1	86.96%
$|Z_2^Y|$	3.9×10^2	86.70%
$|Z_3^Y|$	4.1×10^3	49.89%
$|Z_4^Y|$	1.5×10^4	10.00%
$|Z_5^Y|$	2.8×10^4	1.68%

To generate the data used in our experiments we used the PFAM dataset [El-Gebali et al., 2018]. This dataset consists of aminoacid sequences pertaining to different protein families. To create the data used in our experiments we generate the unlabeled set U (representing the domain of the binary prediction task) by sampling sequence segments of size 5. There are a total of 23 amino acid symbols.

The original sequences are significantly longer than these segments but to derive a more challenging binary classification task, we decided to work with segments. The reason is that deep-models have



(a) MICRO MSPC



(b) MICRO MSPC-U

(c) MSPC, 400 Training Sequences.

	WFA	LSTM	BERT	BASELINE
N1	0.928	0.877	0.888	0.888
N2	0.861	0.726	0.815	0.621
N3	0.645	0.414	0.573	0.167
N4	0.342	0.220	0.304	0.051
N5	0.142	0.106	0.131	0.035
MC	0.161	0.117	0.147	0.037

(d) MSPC-U, 400 Training Sequences.

	WFA	LSTM	BERT
N2	0.772	0.733	0.818
N3	0.633	0.402	0.560
N4	0.340	0.217	0.301
N5	0.142	0.105	0.130
MC	0.158	0.114	0.144

(e) MSPC, 4000 Training Sequences.

	WFA	LSTM	BERT	BASELINE
N1	0.978	0.968	0.964	0.926
N2	0.945	0.878	0.922	0.799
N3	0.730	0.633	0.687	0.44
N4	0.375	0.348	0.358	0.167
N5	0.155	0.149	0.150	0.105
MC	0.176	0.168	0.170	0.112

(f) MSPC-U, 4000 Training Sequences.

	WFA	LSTM	BERT
N2	0.761	0.712	0.755
N3	0.643	0.533	0.591
N4	0.357	0.326	0.336
N5	0.151	0.143	0.145
MC	0.168	0.158	0.160

Figure 1: MSPC and MSPC-U results

been shown to be very successful at classifying the complete sequences into protein families [Bileschi et al., 2019] [Szalkai and Grolmusz, 2018].

To create our target class Y we picked a protein-family and selected all segments that appear in sequences from that family. We selected as our target the most rare or sparse family based on the ratio $|Y|/|U|$. We also selected a second family and created another set Y_{val} to be used as validation data using the same criteria. We created 5 labeled training sets $T = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ by sampling sequences with $m \in \{400, 1000, 2000, 4000, 8000\}$.

4.2 Optimization Details

For each of our training sets $T^m = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ we train a new model and validate the parameters independently. We use a 2 layer LSTM and we validate all parameters such as the dimension of the hidden state, of the embeddings and the learning rate with a grid search evaluated on the validation data Y_{val}. We proceed analogously for the BERT parameters. We use Adam [Kingma and Ba, 2014] as optimizer employing the standard cross entropy loss function. For both LSTM and BERT we train 5 models with different starting random seeds for each training set and we report the average of the results.

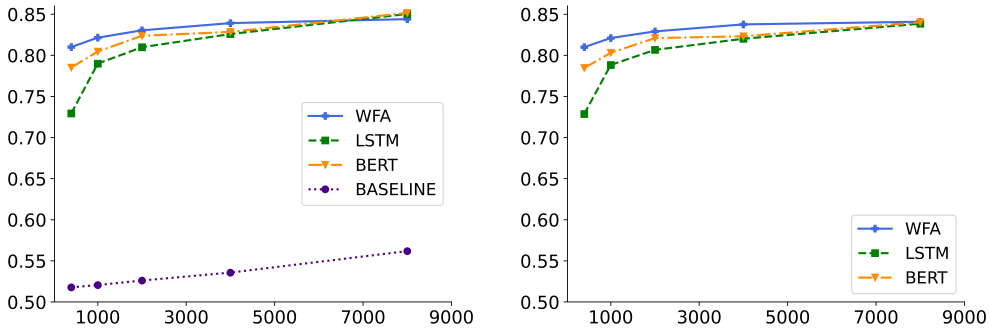
For the WFA model the only parameter that we need to validate is the number of states k. Besides, as there are no randomly initialized parameters and the training is not stochastic, we don't need to perform multiple runs with multiple seeds. The WFA is trained using the spectral learning method.

Table 2: Training time and minimum number of parameters to fully compress 8000 sequences.

	Training time (s)	**# of parameters**
WFA	67	1.15×10^5
LSTM	63	1×10^6
BERT	305	1.1×10^8

For a detailed and comprehensive explanation of the training method the reader is referred to Balle et al. [2014]. Furthermore, we employ the scalabiliy techniques described in Quattoni et al. [2017].

LSTM and BERT models were trained on a single GPU while the WFA, for which the bottleneck is computing a singular value decomposition of a large sparse matrix, was trained on a CPU. To given an idea of the relative size of each model, we computed the minimum number of parameters for which the models were able to achieve perfect classification on our biggest training set. We report the numbers in Table 2.



(a) MICRO MR

(b) MICRO MR-U

(c) MR, 2000 Training Sequences.

	WFA	**LSTM**	**BERT**	**BASELINE**
N4	0.810	0.775	0.807	0.543
N5	0.832	0.812	0.825	0.524
MC	0.830	0.809	0.823	0.526

(d) MR-U, 2000 Training Sequences.

	WFA	**LSTM**	**BERT**
N4	0.809	0.771	0.804
N5	0.830	0.809	0.822
MC	0.829	0.806	0.820

(e) MR, 8000 Training Sequences.

	WFA	**LSTM**	**BERT**	**BASELINE**
N4	0.822	0.818	0.825	0.597
N5	0.845	0.853	0.853	0.558
MC	0.843	0.850	0.851	0.561

(f) MR-U, 8000 Training Sequences.

	WFA	**LSTM**	**BERT**
N4	0.820	0.804	0.812
N5	0.842	0.841	0.842
MC	0.840	0.838	0.840

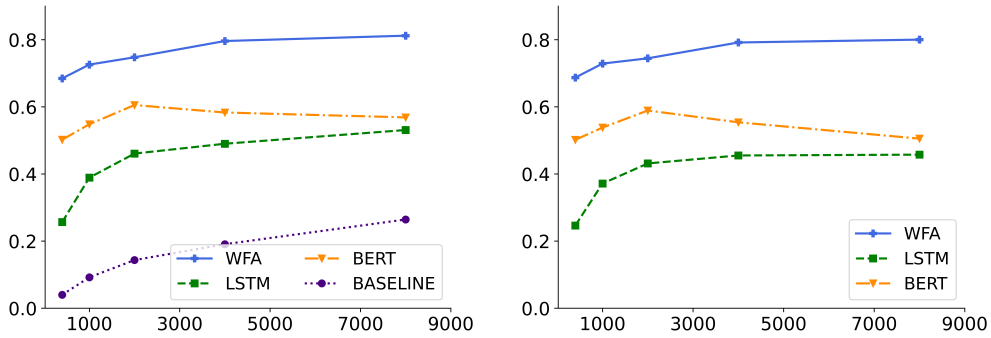
Figure 2: MR and MR-U results

4.3 Results

We now present the results obtained in our experiments, we will break down the evaluation metrics by different *moment* lengths n.

In addition, to summarize the results, for each training set size and model, we compute an aggregate metric, which we refer as MICRO (MC). The MICRO score is defined as a weighted average of the scores for *moments* of different lengths, were the weight is proportional to the total number of test-moments of that size.

Figure 1a shows MICRO MSPC for the three models and the baseline. Tables 1c and 1e show MSPC for different *moment*lengths and fixed training set sizes. The first observation is that all models



(a) MICRO MSPCP

(b) MICRO MSPCP-U

(c) MSPCP 1000, Training Sequences.

	WFA	LSTM	BERT	BASELINE
N1	0.915	0.909	0.930	0.851
N2	0.844	0.717	0.813	0.676
N3	0.692	0.489	0.610	0.262
N4	0.663	0.388	0.518	0.120
N5	0.763	0.369	0.550	0.042
MC	0.725	0.388	0.548	0.091

(d) MSPCP-U, 1000 Training Sequences.

	WFA	LSTM	BERT
N2	0.563	0.633	0.618
N3	0.662	0.438	0.563
N4	0.656	0.377	0.509
N5	0.762	0.366	0.549
MC	0.728	0.371	0.538

(e) MSPCP, 4000 Training Sequences.

	WFA	LSTM	BERT	BASELINE
N1	0.968	0.953	0.947	0.888
N2	0.922	0.831	0.886	0.707
N3	0.770	0.589	0.675	0.465
N4	0.735	0.480	0.565	0.243
N5	0.831	0.475	0.574	0.113
MC	0.796	0.489	0.583	0.190

(f) MSPCP-U, 4000 Training Sequences

	WFA	LSTM	BERT
N2	0.528	0.581	0.543
N3	0.692	0.458	0.560
N4	0.717	0.440	0.531
N5	0.826	0.461	0.563
MC	0.791	0.455	0.553

Figure 3: MSPCP and MSPCP-U results

perform significantly better than the baseline. Recall, that the baseline is essentially remembering the *moments* that it has observed at training, therefore it is natural that as we increase the number of training sequences its performance approaches that of the learnt models.

Figure 1b and Tables 1d and 1f show the analogous results when SPC is computed over unseen *moments*, the baseline in this case is undefined since by definition the expectation over all unseen *moments* will be zero. As to be expected we see that the SPC is lower, but still the models seem to be learning to rank unseen *moments*, and therefore they are making proper generalizations.

However, one obvious observation is that the SPC is not very high in absolute terms (both in the all and unseen scenarios). Looking at the absolute value of the SPC might be misleading because we are modeling a sparse sequence classification task. In other words, the gold expectation for most of the *moments* in the domain is zero (see Table 1) and in this case the SPC metric might be too harsh. Thus, as described in Section 3.3, we turn our attention to MR.

The MR metric reflects the model's ability to predict the non-zero *moments* of the distribution and as shown in Figure 2a (Tables 2c and 2e) all models learn to do this relatively well. This is also true for the generalization setting (Figure 2b and Tables 2d and 2e) in which models are forced to make predictions on unseen *moments*.

To gain more insight into the distributions induced by the models, another way of breaking down the SPC metric is to focus on how well a model ranks the non-zero *moments* of the gold distribution. That is, in this case, we tell the model which *moments* are non-zero and let the model rank them.

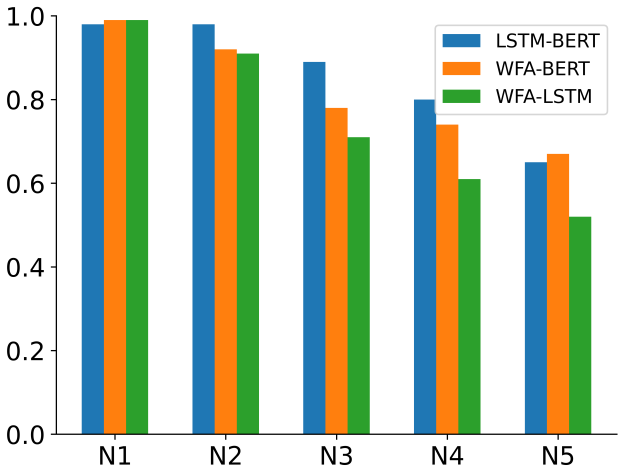


Figure 4: MSPC between models.

Figure 3a and Tables 3c and 3e show MSPCP. We can see that all models learn to rank the *moments* of the target class and this also holds true for unseen *moments* (Figure 3b and Tables 3d 3f). In this evaluation scenario, the WFA seems to be better than the other two models. This is most likely due to the difference between the loss function minimized by the WFA and the loss functions minimized by the LSTM and BERT models. The WFA classifier is defined as an ensemble of class specific models, and therefore it is explicitly modeling the class distributions.

Finally, we look at correlations between different models, to do so we compute pairwise spearman correlation among the model's predictions. Figure 4 shows that the correlation is high for shorter *moments* and that it decreases significantly with longer ones. Regarding pairwise associations, LSTM and BERT show a higher level of agreement, while the WFA is more correlated to BERT overall.

4.4 Conclusion

In this paper we presented a novel evaluation framework for comparing sequence classification models and for testing the extent to which they can implicitly learn the target distribution. Our results comparing three different deep-learning models suggest that the models have the ability to learn the underlying distribution. They also suggest that despite significant differences in the architectures and even in the loss functions (i.e. for the WFA), their performance is comparable.

The goal of this paper was to develop a simple way to gain insights into what is actually been learnt by a given sequence classifier. But the evaluation approach itself also suggests ways in which we could exploit unlabeled data to obtain better calibrated sequence classifiers, future work should explore this possibility.

References

Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are transformers more robust than cnns? *CoRR*, abs/2111.05464, 2021. URL `https://arxiv.org/abs/2111.05464`.

B. Balle, X. Carreras, F. M. Luque, and A. Quattoni. Spectral learning of weighted automata. *Machine Learning*, 96(1):33–63, Jul 2014. ISSN 1573-0565. doi: 10.1007/s10994-013-5416-x. URL `https://doi.org/10.1007/s10994-013-5416-x`.

M. L. Bileschi, D. Belanger, D. Bryant, T. Sanderson, B. Carter, D. Sculley, M. A. DePristo, and L. J. Colwell. Using deep learning to annotate the protein universe. *bioRxiv*, 2019. doi: 10.1101/626507. URL `https://www.biorxiv.org/content/early/2019/05/04/626507`.

N. De Cao, W. Aziz, and I. Titov. Block neural autoregressive flow. In R. P. Adams and V. Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115

of *Proceedings of Machine Learning Research*, pages 1263–1273. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/de-cao20a.html.

S. Desai and G. Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.21. URL https://aclanthology.org/2020.emnlp-main.21.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, E. L. Sonnhammer, L. Hirsh, L. Paladin, D. Piovesan, S. C. Tosatto, and R. D. Finn. The Pfam protein families database in 2019. *Nucleic Acids Research*, 47(D1): D427–D432, 10 2018. ISSN 0305-1048. doi: 10.1093/nar/gky995. URL https://doi.org/10.1093/nar/gky995.

E. Eskin. *Sparse sequence modeling with applications to computational biology and intrusion detection*. Columbia University, 2002.

R. Fakoor, P. Chaudhari, J. Mueller, and A. J. Smola. Trade: Transformers for density estimation. *CoRR*, abs/2004.02441, 2020. URL https://arxiv.org/abs/2004.02441.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016. URL http://arxiv.org/abs/1602.02410.

S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456, 2019. doi: 10.1109/ASRU46091.2019.9003750.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, and X. Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

A. Kumar and S. Sarawagi. Calibration of encoder decoder models for neural machine translation. *CoRR*, abs/1903.00802, 2019. URL http://arxiv.org/abs/1903.00802.

S. M. Lakew, M. Cettolo, and M. Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 641–652, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1054.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102430. URL https://doi.org/10.1145/1102351.1102430.

J. B. Oliva, A. Dubey, M. Zaheer, B. Póczos, R. Salakhutdinov, E. P. Xing, and J. Schneider. Transformation autoregressive networks, 2018.

F. Pinto, P. Torr, and P. K. Dokania. Are vision transformers always more robust than convolutional neural networks? In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. URL https://openreview.net/forum?id=CSXa8LJMttt.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3), 1999.

A. Quattoni and X. Carreras. Interpolated spectral NGram language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5926–5930, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1594. URL https://aclanthology.org/P19-1594.

A. Quattoni and X. Carreras. A comparison between CNNs and WFAs for sequence classification. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 159–163, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sustainlp-1.21. URL https://aclanthology.org/2020.sustainlp-1.21.

A. Quattoni, X. Carreras, and M. Gallé. A Maximum Matching Algorithm for Basis Selection in Spectral Learning. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1477–1485. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/quattoni17a.html.

G. Rabusseau, T. Li, and D. Precup. Connecting weighted automata and recurrent neural networks through spectral learning. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1630–1639. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/rabusseau19a.html.

B. Szalkai and V. Grolmusz. Near perfect protein multi-label classification with deep neural networks. *Methods*, 132:50–56, 2018.

B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *J. Mach. Learn. Res.*, 17(1):7184–7220, jan 2016. ISSN 1532-4435.

W. Yin, K. Kann, M. Yu, and H. Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017. URL http://arxiv.org/abs/1702.01923.