BowTie – A deep learning feedforward neural network for sentiment analysis

Apostol Vassilev
Computer Security Division
Information Technology Laboratory

April 22, 2019

This publication is available free of charge from:
https://doi.org/10.6028/NIST.CSWP.04222019

National Institute of
Standards and Technology
U.S. Department of Commerce

BowTie - A deep learning feedforward neural network for sentiment analysis

Apostol Vassilev

National Institute of Standards and Technology

apostol.vassilev@nist.gov

Abstract

How to model and encode the semantics of human-written text and select the type of neural network to process it are not settled issues in sentiment analysis. Accuracy and transferability are critical issues in machine learning in general. These properties are closely related to the loss estimates for the trained model. I present a computationally-efficient and accurate feedforward neural network for sentiment prediction capable of maintaining low losses. When coupled with an effective semantics model of the text, it provides highly accurate models with low losses. Experimental results on representative benchmark datasets and comparisons to other methods[1]show the advantages of the new approach.

Keywords: machine learning, deep learning, artificial intelligence

Introduction

When approaching the problem of applying deep learning to sentiment analysis one faces at least five classes of issues to resolve. First, what is the best way to encode the semantics in natural language text so that the resulting digital representation captures well the semantics in their entirety and in a way that can be processed reliably and efficiently by a neural network and result in a highly accurate model? This is a critically important question in machine learning because it directly impacts the viability of the chosen approach. There are multiple ways to encode sentences or text using neural networks, ranging from a simple encoding based on treating words as atomic units represented by their rank in a vocabulary [3], to using word embeddings or distributed representation of words [13], to using sentence embeddings. Each of these encoding types have different complexity and rate of success when applied to a variety of tasks. The simple encoding method offers simplicity and robustness. The usefulness of word embeddings has been established in several application domains, but it is still an open question how much better it is than simple encoding in capturing the entire semantics of the text in natural language processing (NLP) to provide higher prediction accuracy in sentiment analysis. Although intuitively one may think that because word embeddings do capture some of the semantics contained in the text this should help, the available empirical test evidence is inconclusive. Attempts to utilize sentence embeddings have been even less successful [4].

Second, given an encoding, what kind of neural network should be used? Some specific areas of applications of machine learning have an established leading network

[1]DISCLAIMER: This paper is not subject to copyright in the United States. Commercial products are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

type. For example, convolutional neural networks are preferred in computer vision. However, because of the several different types of word and sentence encoding in natural language processing (NLP), there are multiple choices for neural network architectures, ranging from feedforward to convolutional and recurrent neural networks.

Third, what dataset should be used for training? In all cases the size of the training dataset is very important for the quality of training but the way the dataset is constructed and the amount of meta-data it includes also play a role. For example, the Keras IMDB Movie reviews Dataset [10] (KID) for sentiment classification contains human-written movie reviews. A larger dataset of similar type is the Stanford Large Movie Review Dataset (SLMRD) [1]. I consider KID and SLMRD in detail in Sections 1.3 and 1.4. Generally, simpler encodings and models trained on large amounts of data tend to outperform complex systems trained on smaller datasets [13].

Fourth, what kind of training procedure should be employed - supervised or unsupervised? Traditionally, NLP systems are trained on large unsupervised corpora and then applied on new data. However, researchers have been able to leverage the advantages of supervised learning and transfer trained models to new data by retaining the transfer accuracy [4].

Fifth, when training a model for transfer to other datasets, what are the model characterizing features that guarantee maintaining high/comparable transfer accuracy on the new dataset? Certainly, training and validation accuracy are important but so are the training and validation losses. Some researchers argue that the gradient descent method has an implicit bias that is not yet fully understood, especially in cases where there are multiple solutions that properly classify a given dataset [14]. Thus, it is important to have a neural network with low loss estimates for a trained model to hope for a good and reliable transfer accuracy.

The primary goal of this paper is to shed light on how to address these issues in practice. To do this, I introduce a new feedforward neural network for sentiment analysis and draw on the experiences from using it with two different types of word encoding: a simple one based on the word ranking in the dataset vocabulary; the other judiciously enhanced with meta-data related to word polarity. The main contribution of this paper is the design of the BowTie neural network in Section 2.

1 Data encoding and datasets

As discussed above, there are many different types of encodings of text with different complexity and degree of effectiveness. Since there is no convincing positive correlation established in the literature between complexity of the encoding and higher prediction accuracy, it is important to investigate the extent to which simple data encodings can be used for sentiment analysis. Simpler encodings have been shown to be robust and efficient [3]. But can they provide high prediction accuracy in sentiment analysis?

I investigate this open question by evaluating the accuracy one may attain using two types of text encoding on representative benchmark datasets.

1.1 Multi-hot encoding

The first encoding is the *multi-hot* encoding of text [6] which can be defined as follows.

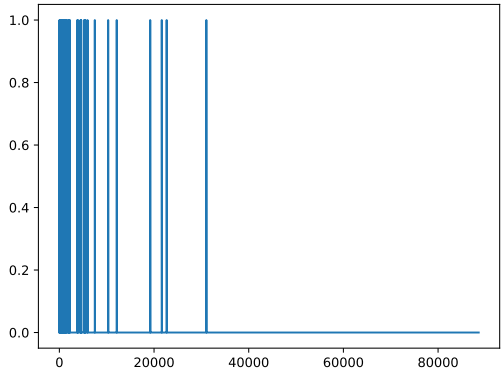

Figure 1. A multi-hot encoded text in D, Π_D with $M = 88\,587$.

Let π be a linguistic type (e.g. morpheme, word) and let Π_D be the set of all such linguistic types in a dataset D. Let $M = |\Pi_D|$ be the cardinality of Π_D. Let ψ be a linguistic text type (e.g., a movie review) and let Ψ_D be the set of all texts in D. Let $N = |\Psi_D|$ be the cardinality of Ψ_D. Let Π_D and Ψ_D be finite sets such that the elements in each set are enumerated by $\{0, ..., M\}$ and $\{0, ..., N\}$ respectively. Let T^{NxM} be a tensor of real numbers of dimensions N by M, whose elements are set as follows:

$$\{\tau_{jk}\} = \begin{cases} 1, & \text{if } \pi_k \in \psi_j; \\ \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The multi-hot encoding (1) represents a very simple model of the semantics in ψ, $\forall \psi \in \Psi_D$. An example of a multi-hot encoded text is shown in Figure 1.

1.2 Polarity-weighted multi-hot encoding

The second encoding I consider is similar to the multi-hot encoding in the sense it has the same non-zero elements but their values are weighted by the cumulative effect of the polarity of each word present in a given text π, as computed by [19]. Let $c_{\pi,\psi}$ be the number of tokens of the linguistic type π in a text ψ. Let ξ_π be the polarity rating of the token $\pi \in \Pi_D$. Naturally, I assume that if Ξ_D is the set of all polarity ratings for tokens $\pi \in \Pi_D$, then $|\Xi_D| = |\Pi_D|$. Let $\omega_{\xi\pi\psi} = \xi_\pi * c_{\pi,\psi}$ be the cumulative polarity of π in the text ψ. Let $\Omega_D = \{\omega_i\}_{i=0}^{M}$ and $C_D = \{c_i\}_{i=0}^{M}$.

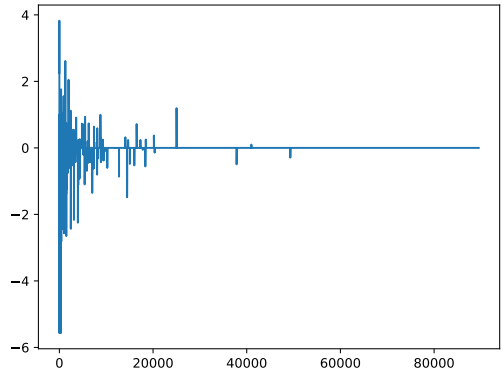

Figure 2. A polarity-weighted multi-hot encoded text in D, Π_D with $M = 89\,527$.

Let Θ^{NxM} be a tensor of real numbers of dimensions N by M, whose elements are set as follows:

$$\{\theta_{jk}\} = \begin{cases} \omega_{\xi_k \pi_k \psi_j}, & \text{if } \pi_k \in \psi_j; \\ \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The polarity-weighted multi-hot encoding (2) represents a more comprehensive model of the semantics in ψ, $\forall \psi \in \Psi_D$, that captures more information about ψ. I will attempt to investigate if and how much this additional information helps to improve the sentiment predictions in Section 3. An example of a polarity-weighted multi-hot encoded text is shown in Figure 2.

1.3 The Keras IMDB Dataset (KID)

The KID [10] contains 50 000 human-written movie reviews that are split in equal subsets of 25 000 for training and testing and further into equal categories labeled as

positive or negative. For convenience, the reviews have been pre-processed and each review is encoded as a sequence of integers, representing the ranking of the corresponding word in Π_D with $|\Pi_D| = 88\,587$. As such, it can be easily encoded by the multi-hot encoding (1).

1.4 The Stanford Large Movie Review Dataset (SLMRD)

SLMRD contains 50 000 movie reviews, 25 000 of them for training and the rest for testing. The dataset comes also with a processed bag of words and a word polarity index [1, 12]. SLMRD contains also 50 000 unlabeled reviews intended for unsupervised learning. It comes with a Π_D, polarity ratings Ω_D, and word counts C_D with $|\Omega_D| = |C_D| = |\Pi_D| = 89\,527$.

2 The BowTie[2] feedforward neural network

Figure 3. The classic bow tie. The bow tie originated among Croatian mercenaries during the Thirty Years' War of the 17th century. It was soon adopted by the upper classes in France, then a leader in fashion, and flourished in the 18th and 19th centuries. (Wikipedia)

As discussed above, the ability of the network to provide accurate predictions and maintain low losses is very important for allowing it to transfer to other datasets and maintain the same or higher prediction accuracy as on the training dataset.

I now introduce a feedforward neural network with that criteria in mind. By way of background [5], logistic regression computes the probability of a binary output $\hat{y}_i$ given an input x_i as follows:

$$P(\hat{\mathbf{y}}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} \mathbf{Ber}[\hat{y}_i|\mathbf{sigm}(x_i\mathbf{w})], \quad (3)$$

where **Ber**[ ] is the Bernouli distribution, **sigm**() is the sigmoid function, $\mathbf{w}$ is a vector of weights. The cost function to minimize is $\mathbf{C}(\mathbf{w}) = -\log P(\hat{\mathbf{y}}|\mathbf{X}, \mathbf{w})$. This method is particularly suitable for sentiment prediction. One critical observation is that logistic regression can be seen as a special case of the generalized linear model. Hence, it is analogous to linear regression. In matrix form, linear regression can be written as

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}, \quad (4)$$

where $\hat{\mathbf{y}}$ is a vector of predicted values $\hat{y}_i$ that the model predicts for $\mathbf{y}$, $\mathbf{X}$ is a matrix of row vectors x_i called regressors, $\mathbf{w}$ are the regression weights, and $\boldsymbol{\epsilon}$ is an error that captures all factors that may influence $\hat{\mathbf{y}}$ other than the regressors $\mathbf{X}$.

The gradient descent algorithm used for solving such problems [5] may be written as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \rho^{(k)}\mathbf{g}^{(k)} + \boldsymbol{\epsilon}^{(k)}, \quad (5)$$

where $\mathbf{g}^{(k)}$ is the gradient of the cost function $\mathbf{C}(\mathbf{w})$, $\rho^{(k)}$ is the learning rate or step size, and $\boldsymbol{\epsilon}^{(k)}$ is the error at step k of the iterative process. One error-introducing factor in particular is the numerical model itself and the errors generated and propagated by the gradient descent iterations with poorly conditioned matrices run on a computer with limited-precision floating-point numbers. Even if regularization is used, the specific parameters used to weigh them in the equation (e.g., the L_2-term weight or the dropout

[2]The name is inspired by the classic image of the bow tie - see Figure 3

rate) may not be optimal in practice thus leading to potentially higher numerical error. This is why it is important to look for numerical techniques that can reduce the numerical error effectively. This observation inspires searching for techniques similar to multigrid from numerical analysis that are very effective at reducing the numerical error [2].

The neural network design

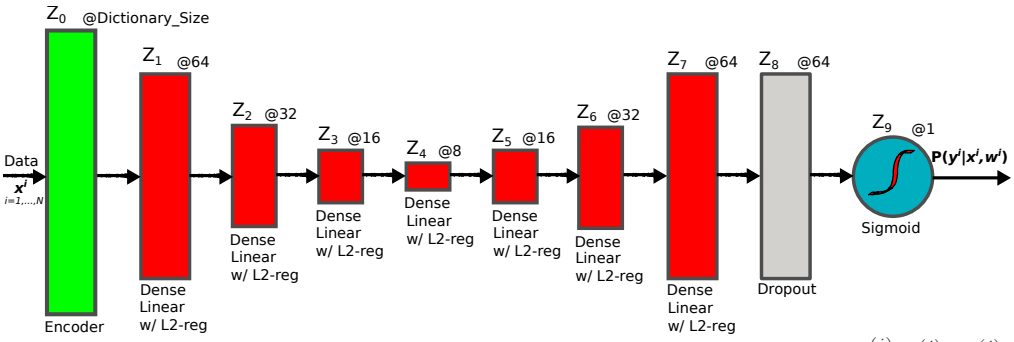

Figure 4. The BowTie neural network. The estimated probability $P(\hat{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}^{(i)})$ may be fed into a post-processing discriminator component to assign a category (pos/neg) for the input $\mathbf{x}^{(i)}$ with respect to a discriminator value $\delta \in [0, 1]$. All experiments presented in this paper use $\delta = 0.5$.

The feedforward neural network in Figure 4 consists of one encoding layer, a cascade of dense linear layers with L_2-regularizers and of appropriate output size followed by a dropout regularizer and a sigmoid. The encoder takes care of encoding the input data for processing by the neural network. In this paper I experiment with the two encodings defined in Section 1: the simple multi-hot encoding and the polarity-weighted multi-hot encoding.

The sigmoid produces the estimated output probability $P(\hat{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}^{(i)})$, which may be used to compute the negative log-loss or binary cross-entropy as

$$-\left[\mathbf{y}\log(P(\hat{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}^{(i)})) + (1-\mathbf{y})\log(1 - P(\hat{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}^{(i)}))\right]. \tag{6}$$

The binary cross-entropy provides a measure for quality and robustness of the computed model. If the model predicts correct results with higher probability, then the binary-cross entropy tends to be lower. If however the model predicts correct results with probability close to the discriminator value or predicts an incorrect category, the binary cross-entropy tends to be high. Naturally, it is desirable to have models that confidently predict correct results. It is also important to have models that maintain low binary cross-entropy for many training epochs because, depending on the training dataset, the iterative process (5) may need several steps to reach a desired validation accuracy. Models that quickly accumulate high cross-entropy estimates tend to overfit the training data and do poorly on the validation data and on new datasets.

Hyperparameters. There are several hyperparameters that influence the behavior of BowTie, see Table 1. For optimal performance the choice of the dense layer activation should be coordinated with the choice for the L_2-regularization weight. This recommendation is based on the computational experience with BowTie and is in line with the findings in [8] about the impact of the activation layer on the training of neural networks in general. The linear network (no dense layer activation) runs better with

Table 1. BowTie hyperparameters.

Hyperparameters	
name	**values/range**
L_2-regularization weight	0.01 - 0.02
Dropout rate	0.2 - 0.5
Optimizer	NADAM, ADAM, or RMSProp
Dense Layer Activation	None (Linear network), RELU

L_2-regularization weight close to 0.02 but rectified linear unit (RELU) activation runs better with L_2-regularization weight close to 0.01. The network can tolerate a range of dropout rates but a dropout rate of 0.2 is commonly recommended in the literature and works well here too. The choice of the optimizer can affect the learning rate, the highest accuracy attained and the stability over several epochs. BowTie performs well with adaptive momentum (ADAM), Nesterov adaptive momentum (NADAM) and Root Mean Square Propagation (RMSPRop), no dense layer activation, L_2-regularization set to 0.019 (this value resulting from hyperparameter optimization) and a dropout rate of 0.2. It is interesting to note that RMSProp tends to converge faster to a solution and sometimes with a higher validation accuracy than NADAM but the transfer accuracy of the models computed with NADAM tends to be higher than for models computed with RMSPRop. For example, a model trained on SLMRD with validation accuracy of 89.24 %, higher than any of the data in Table 4 below, yielded 91.04 % transfer accuracy over KID, which is lower than the results in Table 4. This experimental finding is consistent over many tests with the two optimizers and needs further investigation in future research to explore the theoretical basis for it.

3 Training and transfer scenarios

This section defines the objectives for the testing of the BowTie neural network shown in Figure 4 in terms of four training and transfer scenarios.

But first it is important to decide on the type of training to employ - supervised or unsupervised. I embark on supervised training based on the findings in [4] about the advantages of supervised training and the availability of corpora of large labeled benchmark datasets [10] and [1].

These are the scenarios to explore:

- **Scenario 1 (Train and validate):** Explore the accuracy and robustness of the BowTie neural network with the simple multi-hot encoding by training and validating on KID.
- **Scenario 2 (Train and validate):** Explore the accuracy and robustness of the BowTie neural network with the simple multi-hot encoding by training and validating on SLMRD.
- **Scenario 3 (Train and validate):** Explore the accuracy and robustness of the BowTie neural network with the polarity-weighted multi-hot encoding by training and validating on SLMRD.
- **Scenario 4 (Train, validate, and transfer):** Explore the transfer accuracy of the BowTie neural network with polarity-weighted multi-hot encoding by training on SLMRD and predicting on KID.

The primary goal of this exploration is to establish some baseline ratings of the properties of the BowTie neural network with the different encodings and compare

against similar results for other neural networks with other types of encoding. This provides a quantitative criteria for comparative judging.

Results

In this section I report the results from executing Scenarios 1-4 from Section 3 using TensorFlow [6], version 1.12, on a 2017 MacBook Pro with 3.1 GHz Intel Core i7 and 16 GB RAM *without* Graphics Processing Unit (GPU) acceleration. The test code is written in Python 3 and executes under a Docker image [7] configured with 10 GB of RAM and 4 GB swap.

Scenario 1. In this test, the BowTow neural network is tested with encoding 1. The results in Table 2 show high accuracy and low binary cross-entropy estimates.

Table 2. Scenario 1 results. Data from experiments with training a model on KID until it attains some validation accuracy grater than 88 %. Note that each time the data is loaded for training, it is shuffled randomly, hence the small variation in computational results.

Training and validating on KID	
validation accuracy (%)	validation binary cross-entropy
88.08	0.2955
88.18	0.2887
88.21	0.2945

To assess the relative computational efficiency of the BowTie neural network, I compared it to the convolutional neural network in [11] with a 10 000 word dictionary. The network reached accuracy of 88.92 % at Epoch 4 with binary cross entropy of 0.2682. However, it took 91 seconds/Epoch, which matched the numbers reported by the authors for the CPU-only computational platform. In addition, after Epoch 4, the binary cross-entropy started to increase steadily while the accuracy started to decline. For example, the binary cross-entropy reached 0.4325 at Epoch 10 with validation accuracy of 87.76 % and 0.5536 and 87.40 % correspondingly at Epoch 15.

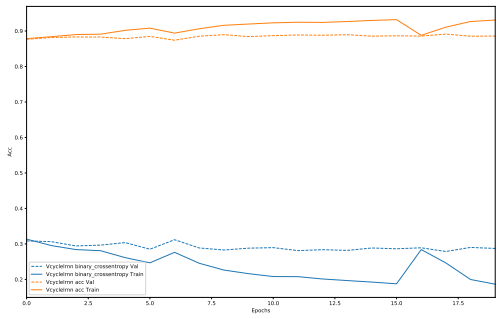

Figure 5. BowTie accuracy and cross-entropy results. The neural network keeps the cross-entropy estimate low over the course of 20 epochs.

In comparison, BowTie took only 3 seconds/Epoch for the same dictionary size and attained accuracy of 88.20 % with binary cross-entropy of 0.2898. The binary cross-entropy stays stable below 0.38 for a number of Epochs. In other words, BowTie is 30-times faster, attains comparable accuracy and maintains stable binary-cross entropy.

Scenario 2 SLMRD is more challenging than KID for reasons that are visible in the test results for Scenarios 3 and 4, hence the slightly lower validation accuracy attained by BowTie using the simple encoding 1 - it easily meets or exceeds the threshold accuracy of 87.95 % but could not surpass the 88 % limit in several experiments.

Table 3. Scenario 2 results. Data from experiments with training a model on KID until it attains some validation accuracy was equal to or grater than 87.95 %. Note that each time the data is loaded for training, it is shuffled randomly, hence the small variation in computational results.

Training and validating on SLMRD	
validation accuracy (%)	validation binary cross-entropy
87.98	0.2959
87.95	0.2996
87.96	0.3001

Scenarios 3 and 4. I combine the reporting for Scenarios 3 and 4 because once the model is trained under Scenario 3 it is then transferred to compute predictions on KID. To perform the transfer testing on KID one needs to reconcile the difference in $|\Pi_{SLMRD}|$ and $|\Pi_{KID}|$. As I noted in Section 1, $|\Pi_{SLMRD}| = 89\,527$ and $|\Pi_{KID}| = 88587$. Moreover, $\Pi_{KID} \not\subset \Pi_{SLMRD}$. Let $\Pi_{\Delta} = \Pi_{KID} \setminus (\Pi_{KID} \cap \Pi_{SLMRD})$. It turns out that

$$\Pi_{\Delta} = \left\{ \begin{array}{l} 0s, 1990s, 5, 18th, 80s, 90s, 2006, 2008, \\ 85, 86, 0, 5, 10, tri, 25, 4, 40s, 70s, 1975, \\ 1981, 1984, 1995, 2007, dah, walmington, \\ 19, 40s, 12, 1938, 1998, 2, 1940's, 3, 000, 15, 50. \end{array} \right\}.$$

Clearly, $|\Pi_{\Delta}|$ is small and of the words in Π_{Δ}, only 'walmington' looks like a plausible English word but it is the name of a fictional town in a British TV series from the 1970's. As such it has no computable polarity index and is semantically negligible. Based on this, I dropped all these words during the mapping of $\pi \in \Pi_{\Delta}$ into the corresponding $\pi\prime \in \Pi_{SLRMD}$. Note that this $\pi \to \pi\prime$ mapping enables the encoding of the semantics of the texts in KID according to 2.

Some simple but revealing statistics about the data. With encoding 2, the matrix T_{SLMRD} for the training set in SLMRD shows cumulative polarity in the range [-50.072 837, 58.753 546] and the cumulative polarity of the elements of the matrix T_{SLMRD} for the test set is in the range [-48.960 107, 63.346 164]. This suggests that SLMRD is pretty neutral and an excellent dataset for training models. In contrast, the elements of the matrix T_{KID} are in the range [-49.500 000, 197.842 862].

Table 4. Scenarios 3 and 4 results. Data from experiments with training a model on SLMRD until it attains some validation accuracy grater than 89 % and then using that same model to predict the category for each labeled review in KID. Note that the transfer accuracy is computed over the entire set of 50 000 reviews in KID.

Training on SLMRD		**Predicting on KID**
validation accuracy (%)	validation binary cross-entropy	Prediction transfer accuracy (%)
89.02	0.2791	91.56
89.12	0.2815	91.63
89.17	0.2772	91.76

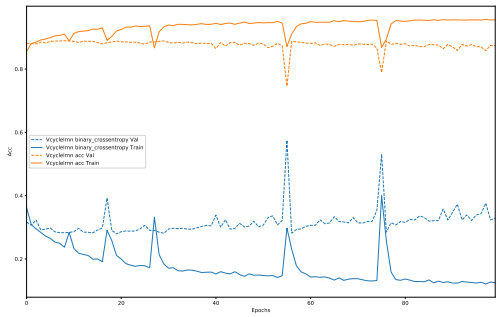

Figure 6. BowTie accuracy and cross-entropy results. The neural network keeps the cross-entropy estimate in check over the course of 100 epochs.

Table 4 contains the results from executing Scenarios 3 and 4. The validation and transfer accuracy results in Table 4 are better than those shown in Tables 2 and 3. This shows the value word polarity brings in capturing the semantics of the text. The transfer accuracy over KID is also higher than the results obtained by using the convolutional neural network [11] on KID.

The results in Table4 are higher than the results reported for sentiment prediction of movie reviews in [4] but also in agreement with the reported experience by these authors of consistently improved accuracy from supervised learning on a large representative corpus before transferring the model for prediction to a corpus of interest.

Note also that the validation accuracy results for the BowTie neural network on SLMRD are substantially higher than the results for the same dataset in [12]. The observation in [12] that even small percentage improvements result in a significant number of correctly classified reviews applies to the data in Table 4: that is, there are between 172 and 210 more correctly classified reviews for BowTie. In addition, BowTie is computationally stable and retains low cross-entropy losses over a large number of epochs, see Figures 5 and 6, which is a desirable property.

The speed of computation improves on platforms with GPU acceleration. For example, experiments on a system *with* eight Tesla V100-SXM2 GPUs yield speedups of nearly a factor of two in training but the acceleration plateaued if more than two GPUs were used. The computational speedup was better during prediction computations with a trained model: the time needed to calculate prediction for the entire set of 50 000 reviews reduced from 44 secs on one GPU to 31 secs on two GPUs, to 28 secs on four GPUs and to 26 secs on eight GPUs.

Discussion and next steps

The experimental results from sentiment prediction presented above show the great potential deep learning has to enable automation in areas previously considered impossible. At the same time, we are witnessing troubling trends of deterioration in cybersecurity that have permeated the business and home environments: people often cannot access the tools they need to work or lose the data that is important to them [20]. Traditionally, governments and industry groups have approached this problem by establishing security testing and validation programs whose purpose is to identify and eliminate security flaws in IT products before adopting them for use.

One area of specific concern in cybersecurity is cryptography. Society recognizes cryptography's fundamental role in protecting sensitive information from unauthorized disclosure or modification. The cybersecurity recommendations in [20] list relying on cryptography as a means of data protection as one of the top recommendations to the business community for the past several years. However, the validation programs to this day remain heavily based on human activities involving reading and assessing human-written documents in the form of technical essays - see Fig. 7 for an illustration of the structure of the Cryptographic Module Validation Program (CMVP) [18] established in 1995 to validate cryptographic modules against the security requirements

in Federal Information Processing Standard (FIPS) Publication 140-2 [15].

This validation model worked well for the level of the technology available at the time when the programs were created more than two decades ago. As technology has advanced, however, this model no longer satisfies current day industry and government operational needs in the context of increased number and intensity of cybersecurity breaches [20].

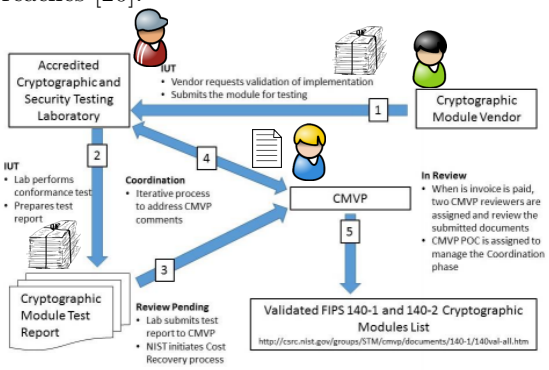

Figure 7. CMVP structure and processes Processes rely entirely on human actors and human-readable artifacts (English essays)

There are several factors for this. First, current cybersecurity recommendations [20] require that every organization relying on technology today patch promptly, including applying patches to cryptographic modules. Technology products are very complex and the cost of testing them fully to guarantee trouble-free use is prohibitively high. As a result, products contain vulnerabilities that hackers and technology producers are competing to discover first: the companies to fix, the hackers to exploit. Patching products changes the game for hackers and slows down their progress. Thus, patching promptly is a way of staying ahead of security breaches. However, patching changes also the environment in which a cryptographic module runs and may also change the module itself, thus invalidating the previously validated configuration. Users who depend on validated cryptography face a dilemma when frequent updates and patches are important for staying ahead of the attackers, but the existing validation process does not permit rapid implementation of these updates while maintaining a validated status because of the slow human-based activities illustrated in Fig. 7.

The second factor hindering the effectiveness of the traditional validation model shown in Fig. 7 is the demand for speed in the context of the cognitive abilities of the human brain. Rapid changes in technology and related steep learning curves are stretching the resources at the testing laboratories. When evaluation package submissions finally reach the validation queue, inconsistent and possibly incomplete evidence presentation further strains the ability for a finite number of reviewers to provide timely turnaround. Recent scientific research points out that humans are limited in their ability to process quickly and objectively large amounts of complex data [9]. Two systems drive the way humans think: the fast, intuitive, and emotional System 1; the slower, more deliberative, and more logical System 2. Systems 1 and 2 constantly interact but System 1 is always in the driver's seat. This leads to faults and biases, of fast thinking, and reveals the pervasive influence of intuitive impressions on human thoughts and behavior. These are the primary reasons why we cannot trust human intuitions when dealing with highly complex data.

Going back to the results on sentiment analysis with deep learning from above and in spite of that success people may always question the ability of machines to replace humans in solving such cognitive and analytical tasks. They will always ask why the accuracy is not one hundred percent? Or, notwithstanding the available scientific evidence [9], say that if a human was reviewing the text she would have never made a mistake.

Changing public opinion may be a slow process. Besides, it seems that cybersecurity and machine learning/artificial intelligence (AI) will always be joined at the hip because the more we rely on machines to solve ever more complex tasks, the higher the risk those machines may be attacked to subvert their operation. Can AI fight back though?

The successful results presented in this paper suggest that computer-based validation of cryptographic test evidence may be the only viable alternative that would allow objective and accurate assessment of large volumes of data at the speed required by the cybersecurity reality [20]. This paper demonstrates that deep learning neural networks are capable of tackling the core tasks in security validations and thereby automate existing programs [17]. If this effort is indeed successful one may reason that by helping to improve the process of validation and thereby increase cybersecurity, albeit indirectly, AI will in fact be defending itself from cybersecurity threats. Over time, this may lead to societal acceptance of AI into sensitive domains such as the validation of critical components for the IT infrastructure.

Next steps. The importance of incorporating word polarity into the model is illustrated clearly by the results presented in this paper. However, the type of language used in the validation test reports tends to be different than the colloquial English used in movie reviews. The technical jargon in test reports uses many common words whose meaning changes in this context. Moreover, the assessments for each test requirement in [16] are written in a way very different from movie reviews. Here the author provides arguments that justify her conclusion about compliance. The challenge is to distinguish weak/faulty arguments from solid ones. Therefore, a new type of polarity needs to be developed. Related to that is the task of assembling a representative corpus of labeled validation test report data for training and validation.

Acknowledgments

I thank the NIST Information Technology Laboratory (ITL) and especially Elham Tabassi for the research funding and support under ITL Grant #7735282-000.

References

1. Andrew Maas. Large movie review dataset. `http://ai.stanford.edu/~amaas/data/sentiment/`, 2011.

2. J. H. Bramble. *Multigrid Methods*. Wiley, 1993. Pitman Research Notes in Mathematics Vol. 294.

3. T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (Prague)*, pages 858–867, June 2007.

4. A. Conneau, D. Kiela, H. Schwenk, L. Barraul, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Emprical Methods in Natural Language Processing (Copenhagen, Denmark, September 7-11))*, Association of Computational Linguistics, pages 670–680, 2017, See also update at `https://arxiv.org/abs/1705.02364v5`.

5. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

6. Google LLC. Tensorflow: An open source machine learning framework for everyone. `https://www.tensorflow.org/`, 2019.

7. Google LLC. Tensorflow: Docker. https://www.tensorflow.org/install/docker, 2019.

8. S. Hayou, A. Doucet, and J. Rousseau. On the impact of the activation function on deep neural networks training. https://arxiv.org/abs/1902.06853, 2019.

9. D. Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011.

10. Keras Documentation. Imdb movie reviews sentiment classification. https://keras.io/datasets/, 2018.

11. Keras Team. Demonstration of the use of convolution1d for text classification. https://github.com/keras-team/keras/blob/master/examples/imdb_cnn.py, 2019.

12. A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

13. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013, see also https://arxiv.org/abs/1310.4546.

14. M. S. Nacson, J. Lee, S. Gunasekar, P. H. P. Savarese, N. Srebro, and D. Soudry. Convergence of gradient descent on separable data. https://arxiv.org/abs/1803.01905v2, 2018.

15. NIST. Security requirements for cryptographic modules, federal infomation processing standard (FIPS) 140-2. https://doi.org/10.6028/NIST.FIPS.140-2, 2001.

16. NIST. Derived test requirements for fips pub 140-2, security requirements for cryptographic modules. https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402DTR.pdf, 2011.

17. NIST. Automated cryptographic validation testing. https://csrc.nist.gov/projects/acvt/, 2018.

18. NIST. Cryptographic Module Validation Program. https://csrc.nist.gov/projects/cryptographic-module-validation-program, 2018.

19. C. Potts. On the negativity of negation. In *Proceedings of Semantics and Linguistic Theory*, volume 20, pages 636–659, 2011.

20. Verizon. 2018 data breach investigations report. https://enterprise.verizon.com/resources/reports/dbir/, 2018.