Adversarial Modeling of Systemic Errors in Event Encoding

SNEHA MEHTA and TALHA GHAFFAR

1 ABSTRACT

Global scale societal event databases such as ICEWS have been used for deploying useful event forecasting systems. Since these databases have existed for a long time the text processing technology used for extracting events is also very old and has led to corruption of the databases with duplicated and erroneous event codings. One of the causes of the errors are misformed sentences that do not fit event extraction templates. In this work, we propose an adversarial framework to rephrase anomalous sentences for successful codings by state-of-the-art event coders.

2 INTRODUCTION

The Integrated Conflict Early Warning System (ICEWS) is a DARPA funded database developed by Lockheed Martin containing political events. The taxonomy of the data gives varying degrees of resolution for event participants, locations and types. Table 1 gives details about some of the different event attributes in the database. Each event in ICEWS is extracted from a news story taken from a specified subset of news sources, and coded into event descriptors. Since it's inception ICEWS has been used for a variety of geopolitical forecasting and crisis prediction systems [2]. Automated event coding works by parsing individual sentences into SUBJECT VERB OBJECT format and categorizes action using a framework like CAMEO (Conflict and Mediation Event Observations) [4]. So a statement like "Secretary of State John Kerry complained about Russia's support of Syria's Bashar al-Assad" would be coded as US GOVERNMENT/DISAPPROVE/RUSSIAN GOVERNMENT.

The text-processing algorithms used in event coding [1] are still similar to the ones developed more than 20 years ago as a result of which the validity and/or reliability of ICEWS events has taken a hit [3]. Wang et. al [4] performed a human study to assess the validity of the protest events in ICEWS by asking the human subjects to validate the occurrence of the events from the source URL from which the ICEWS events had been coded. From the 431,549 events they examined they found that about 20 % were erroneous or duplicated events. See Table 2 for examples of erroneous encodings in ICEWS.

The source of many of the errors can be traced to misaligned sentences that do not fit the event extraction templates. It is clear that keyword filtering based approaches and other text processing algorithms used in state-of-the-art event encoding systems are not sufficient for accurate event encodings and lead to corruption of databases like ICEWS. For this reason advanced text processing algorithms using learning based approaches need to be adopted either to rectify the errors in the 'big' data sources such as ICEWS or new event coding algorithms need to be developed. With this motivation we describe an adversarially regulated encoder/decoder model that can be used as an event encoder or a sentence rephraser.

3 METHODOLOGY

3.1 Dataset

We have described the ICEWS dataset above. For the purpose of the project we selected a large subset of the above data. We selected events from the year 2000 to 2015 and filtered the dataset for only the protest events.

Authors' address: Sneha Mehta, sudo777@vt.edu; Talha Ghaffar, gtalha@vt.edu.

2017. XXXX-XXXX/2017/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Event ID	A unique internal identifier for the event that is not guaranteed to be consistent across different versions of the data. It is included for the purposes of bug reporting only.
Event Date	The date of the event, in YYYY-MM-DD format
Source Name	The name of the actor that is the source of the event.
Target Name	The name of the actor that is the target of the event.
Source Country	The name of the country with which the source actor is associated if known.
Target Country	The name of the country with which the target actor is associated if known
Event Text	The text of the event type as coded by CAMEO
Story ID	A unique internal identifier for the story from which the event is derived
Event Sentence	The sentence in the story from which the event is derived

Table 1. Select ICEWS Events data fields

Source ID Representative Paragraph	Reason for Error
Since its inception, the Islamic State group has demonstrated the firmness of its structure and the strength of its organizational composition.	Presence of word 'demonstrate' results in a false positive.
Capriles has called off a march by his supporters in Caracas, saying that his rivals were plotting to "infiltrate" the rally to trigger violence.	The protest was called off.

Table 2. Erroneous ICEWS events encodings. Courtesy [3]

The event codes corresponding to protest events are given in table **??**. For each event in the resulting dataset we considered sentences from which the events were extracted as our data. We removed all sentences with length greater than 30 words (as tokenized by the nltk toolkit) which resulted in total of 166071 sentences with 30 words or less. We created a 160000/6071 train/test split.

CAMEO Code	Event Text
140	Engage in political dissent, not specified below
141 (1411, 1412, 1413, 1414)	Demonstrate or Rally
142 (1421, 1422, 1423, 1424)	Conduct hunger strike
143 (1431, 1432, 1433, 1434)	Conduct strike or boycott
144 (1441, 1442, 1443, 1444)	Obstruct passage
145 (1451, 1452, 1453, 1454)	Protest violently, riot

Table 3. Protest event codes in ICEWS and their descriptions.

3.2 Model

We employ an approach most similar to [5]. In [5] authors propose an adversarially regulated autoencoder where the autoencoder simultaneously learns by the reconstruction loss from the decoder and the adversarial loss from the critic. Learning on adversarial loss helps it maintain valid code space representations of real inputs (as opposed to invalid or fake inputs) from a discrete space such as text. A distinct advantage of the method is that it is able to reconstruct(decode) noisy inputs correctly as opposed to a regular autoencoder.

Let $X = V^n$ where n is the sentence length and V is the vocabulary. P_x is the distribution of all sentences in our training dataset which are the sentences from which the events were coded. Following the usual practice we use an RNN as both an encoder and the decoder. Define RNN as the parameterized recurrent function $h_j = RNN(x_j, h_{j-1}; \phi)$ for $j = 1..n$ (with $h_0 = 0$) that maps a discrete input sentence x to hidden vectors $h_1 ... h_n$. For the encoder we define $enc_\phi = h_n = c$ the last hidden state in this recurrence [5].

Consider the case when a state-of-the-art event coder incorrectly encodes a sentence. We propose a sentence rephrasing task in which we hope to learn a code space over all sentences from which the events have been correctly encoded. The learned encoder model will be then able to map a bad sentence sample to a sentence in the valid code space. For this task the decoder is defined in a similar way as the encoder. Fig 1 demonstrates the architecture of our model. Sentence samples from our training set are given to the encoder enc_ϕ which then maps them to a code space c. The generator g_θ generates fake code samples $\tilde{c}$. The critic f_w learns to distinguish between the fake and the real samples and the gradients of the loss L_{WGAN} are backpropogated through both the generator and the encoder. Moreover, the reconstruction loss from the decoder is also backpropogated through the encoder. Eventually the autoencoder learns a good code space representation of the input sentence space. Now given a noisy input to the encoder it should be able to map that sentence to a code in the learned code space and the decoder outputs the rephrased sentence.

Sentence rephrasing is useful in the case when an event encoding system has a low confidence for an event given a sentence, the sentence can be reconstructed using the above autoencoder to map it to a sentence which will generate a high confidence event encoding.

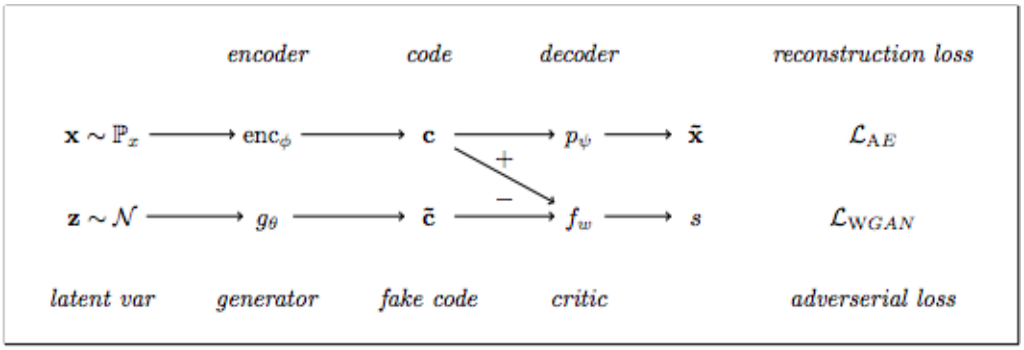

Fig. 1. Model Architecture. Fig Courtesy [5]

We used the training procedure as described in [5] and used a block coordinate descent to alternate between optimizing different parts of the model: (1) the encoder and decoder to minimize reconstruction loss, (2) the WGAN critic function to approximate the W term, (3) the encoder and generator to adversarially fool the critic to minimize W. The training proceeds sequentially for the above three parts, with first the autoencoder trained for reconstruction and backpropogating the resconstruction loss to update autoencoder weights, followed by training the discriminator with fake code samples generated by the generator and real sample generated by

the encoder and finally training the generator and encoder adversarially to the critic that is backpropogating adversarial loss through the generator and the encoder. Complete training algorithm is given in fig. 2.

Algorithm 1 ARAE Training

for number of training iterations **do**

(1) Train the autoencoder for reconstruction $[\mathcal{L}_{\text{rec}}(\phi, \psi)]$.

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$ and compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$.

Backpropagate reconstruction loss, $\mathcal{L}_{\text{rec}} = -\frac{1}{m}\sum_{i=1}^{m} \log p_\psi(\mathbf{x}^{(i)} | \mathbf{c}^{(i)}, [\mathbf{y}^{(i)}])$, and update.

(2) Train the critic $[\mathcal{L}_{\text{cri}}(w)]$ (Repeat k times)

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$ and $\{\mathbf{z}^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})$.

Compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{c}}^{(i)} = g_\theta(\mathbf{z}^{(i)})$.

Backpropagate loss $-\frac{1}{m}\sum_{i=1}^{m} f_w(\mathbf{c}^{(i)}) + \frac{1}{m}\sum_{i=1}^{m} f_w(\tilde{\mathbf{c}}^{(i)})$, update, clip the critic w to $[-\epsilon, \epsilon]^d$.

(3) Train the generator and encoder adversarially to critic $[\mathcal{L}_{\text{encs}}(\phi, \theta)]$

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$ and $\{\mathbf{z}^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})$

Compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{c}}^{(i)} = g_\theta(\mathbf{z}^{(i)})$.

Backpropagate adversarial loss $\frac{1}{m}\sum_{i=1}^{m} f_w(\mathbf{c}^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(\tilde{\mathbf{c}}^{(i)})$ and update.

Fig. 2. Training Algorithm. Courtesy [5]

3.3 Training

We train the ARAE model on 160000 sentences extracted from the ICEWS dataset. We trained the model for 18 epochs using batch size of 64. During the training process, autoencoder and generator was trained for 1 iteration and the discriminator was trained for 5 iterations. At the end of each epoch, the autoencoder(AE) is evaluated with the test data. The AE evaluation measures accuracy of AE sentence generations using test data as input, perplexity of the generated sentences and test loss. We report the AE evaluation results on test data during the training process in Figure 3. The training parameters for the model are described in Table 4.

Training Parameter	**Value**
Epochs	18
Batch Size	64
AE training iterations	1
Generator training iterations	1
Discriminator training iterations	5
AE learning rate	1
Generator learning rate	5e-05
Discriminator learning rate	1e-05
Gradient Clipping	1

Table 4. ARAE Training Parameters

4 EXPERIMENTS AND RESULTS

In table 5 and table 6 we show sample outputs of AE and the generator of the GAN. While training we pruned our vocabulary to contain 11k words, so the <oov> tags correspond to the out of vocabulary words.

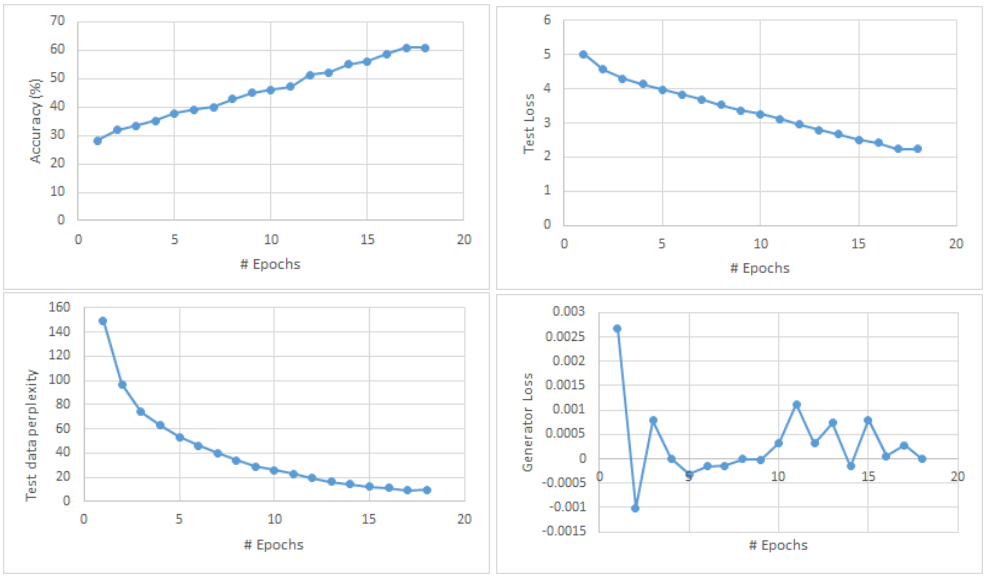

Fig. 3. Training Statistics. **(a)** Autoencoder accuracy on test data during training. **(b)** AE test loss during training. **(c)** AE perplexity of decoded test sentences **(d)** Generator training loss

Original	**Reconstructed**
Thousands have taken to the streets in recent weeks to demand better services and to accuse state officials of <oov> off the country's money.	Thousands have taken to the streets in recent weeks to demand better and and to arrest <oov> <oov> of <oov> the the country's authorities.
The family and relatives staged a protest demonstration on <oov> Road and chanted slogans against police.	The Congress and relatives staged a demonstration demonstration on <oov> Road and chanted slogans against police.

Table 5. ARAE Reconstructed output

Generated
The protesters rallied outside the UN offices on Tuesday demanding immediate <oov> and other bodies of jailed militants from Afghanistan.
The people are demonstrating in the streets of Bangkok to protest the poor conditions of <oov> shouting slogans including <oov>

Table 6. GAN Output

4.1 Sentence Rephrasing

In the ideal case, the ARAE would learn a code space robust to small changes, which is also the main goal of this project. We perform an experiment in which we introduce small noise in the input sentences and measure the average reconstruction error (negative log-likelihood averaged over sentences). Noise is introduce by randomly permuting k words in the sentence. First we filter train sentences by word length and only consider sentences

with less than or equal to 10 words and further filter them to contain only sentences deemed as grammatically incorrect by an open source project language check [1]. This resulted in total 2562 sentences of length 10 or shorter. We compute reconstruction error averaged over these sentences varying the value of k.

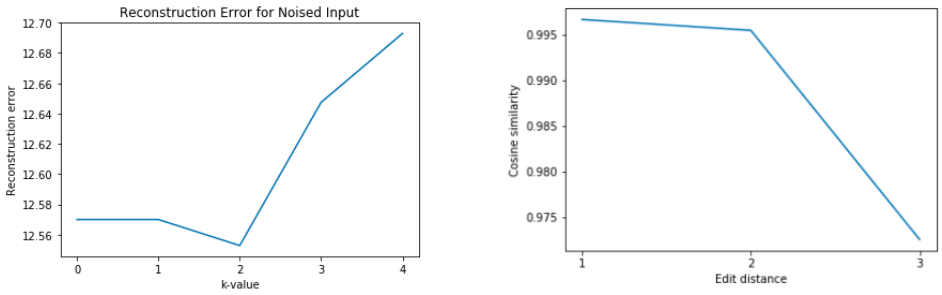

Fig. 4. **Left**: k-value vs Reconstruction error. **Right**: Cosine similarity between two sentences as we vary the edit distance.

Results can be seen in Fig. 4 (left). It is surprising to observe a small dip when $k = 2$, but the results aren't significant. As expected average reconstruction error goes up with increasing value of k. Some of the sample reconstructions can be seen in table 7. An interesting thing about the results is that both of the original sentences have similar structure, only the source and target entities that 'raised full-throated slogans' differ but reconstructions for both are very similar. The noisy and repeated reconstructions can be attributed to paucity in training. Moreover, we plotted the cosine similarity between the code-space representations of the second sentence and it's noised sentence as we vary the edit distance. Fig. 4(right) shows how the cosine similarity gradually decreases with increasing edit distance.

Original	**Permuted**	**Reconstructed**
The protestors raised full-throated anti-India and pro-liberation slogans.	full-throated enraged protesters The raised anti-India and pro-liberation slogans	slogans by by district anti-India and shouted slogans led <eos>
Students raised full-throated slogans against the government.	Students raised full-throated government slogans the against	from the slogans slogans against State Indian State

Table 7. $k = 3$ Permutation and Reconstructed output

We also tried to measure how well the generations try to capture the data distribution by calculating reverse perplexity. To calculate reverse perplexity, we trained an N-gram language model on the generated sentences with and measured the perplexity on test data. For one experiment, we trained 5-gram LM using 10, 000 sentences; and measured perplexity on test data to be 602.56, which is quite high. For comparison, we trained the same 5-gram LM with real-data (training data), the perplexity on test data with this trained model was 234.01. In our experiment, one difference from [5] is that they reported reverse perplexity results using an RNN language model whereas we used a simple 5-gram LM. This could be a contributing factor to the observed high reverse-PPL values in our experiment. We also observed that by training on increased

Generated Sentences	**Reverse PPL**
1000	481.17
5000	559.38
10,000	602.56
100,000	780.91

Table 8. Reverse Perplexity

[1] https://github.com/myint/language-check

number of generated sentences, the reverse perplexity increased because the N-gram model is not able to capture a very large distribution.

4.2 Sentence Interpolation

We ran an experiment to interpolate between the text generated by the generator of the GAN trained with the ARAE method. The experiment involved sampling two points in z-space, generating their linear interpolations and feeding the points to the trained generator to generate code space representation. The generated hidden representation is then passed through AE's decoder to generate the output text. The results of this experiment are shown in table 9. The first sentence in each column is generated by sampling $z1$ from the noise prior and the last is generated using $z2$. The sentences 2, 3, 4 are generated using linear interpolations of $z1$ and $z2$.

People to the school protested the streets and later on Friday of Ramadan.	Around 200 demonstrators entered the High Court and raised slogans against him.
According to the various parts of the area on Thursday, supporters and students protested.	At least 5,000 women had blocked off by protesting corruption in her release.
According to various the number of people took to the city streets and raised slogans.	At least 5,000 women went to a court of <oov> arrested 10 protests which started on Wednesday.
According to reports, a large number of people took to street and protested outside.	At least 2,000 women marched on a court of <oov> area outside to protest leaders were going on Wednesday.
According to him, a large number of people rallied and shouted slogans	A police officer of students gathered at a <oov> while gathering on <oov> Road to be closed away of about 30 minutes.

Table 9. Sample text interpolations. The text is constructed by linearly interpolating the noise priors to generate linear interpolations in latent space. The generated latent space representation is then decoded to produce the final output text.

We ran another experiment similar to above where multiple noise priors $z_0, z_1, z_2, ..., z_n$ are sampled and normalized. The first sentence is generated using z_0. To generate other sentences, small noise is added to z_0 by scaling down other noise priors and adding to z_0. The resulting text that is generated by ARAE-GAN shows this effect in the learned latent space. The results of this experiment are shown in table 10

They will be demonstrated before the state and are boycotting the vote.	The demonstration comes after millions of demonstration in the province.
They will be demonstrated **because** the **government** and **made** the **decision**.	The demonstration comes after millions of demonstration in the **Czech region.**
They will be demonstrated before the **decision**.	The demonstration comes after millions of demonstration in the **South Bank.**
They will be demonstrated before the **Japanese** and **waved banners** the **<oov>**	The demonstration comes after millions of **workers participated in the occupied West**
They will be demonstrated **and he criticized** the **people**.	The demonstration comes after **in January 22** of the **Islamic Revolution.**
They will **stage march and demanded** the **government quit**.	The demonstration comes after in the **Iranian authorities.**

Table 10. The latent space representation for first sentence in each column is generated using noise prior z_0. Other sentences are generated by adding small noise to z_0. The hidden space generations are then decoded using the AE's decoder to generate the text shown in above figure.

5 CONCLUSION

We trained an adversarially regularized autoencoder, an approach for training a discrete structure autoencoder jointly with a code-space generative adversarial network on the ICEWS events dataset. We ran multiple experiments for testing the distribution learnt by the generator, the continuous code space representation of text learnt by the autoencoder and results of iterpolations in the code space and resulting text generations. From our results we conclude that this method is indeed invaluable in learning meaningful code space representations for discrete structures such as text, but needs a lot of training.

REFERENCES

[1] E. Boschee, P. Natarajan, and R. Weischedel. Automatic extraction of events from open source text for predictive forecasting. *Handbook of Computational Approaches to Counterterrorism*.

[2] N. Ramakrishnan, P. Butler, S. Muthiah, N. Self, R. Khandpur, P. Saraf, W. Wang, J. Cadena, A. Vullikanti, G. Korkmaz, C. Kuhlman, A. Marathe, L. Zhao, T. Hua, F. Chen, C. T. Lu, B. Huang, A. Srinivasan, K. Trinh, L. Getoor, G. Katz, A. Doyle, C. Ackermann, I. Zavorin, J. Ford, K. Summers, Y. Fayed, J. Arredondo, D. Gupta, and D. Mares. 'beating the news' with embers: Forecasting civil unrest using open source indicators. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1799–1808, New York, NY, USA, 2014. ACM.

[3] P. Saraf and N. Ramakrishnan. Embers autogsr: Automated coding of civil unrest events. *KDD 2016*.

[4] W. Wang, R. Kennedy, D. Lazer, and N. Ramakrishnan. Growing pains for global monitoring of societal events. *Science Magazie, pages 1502:1503, 2016*.

[5] J. J. Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun. Adversarially regularized autoencoders for generating discrete structures. *CoRR*, abs/1706.04223, 2017.