Learning Semantic Coherence for Machine Generated Spam Text Detection

Mengjiao Bao, Jianxin Li, Jian Zhang, Hao Peng, Xudong Liu
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
SKLSDE Lab, Beihang University
Beijing, China
baomj, lijx, zjian, penghao, liuxd@act.buaa.edu.cn

***Abstract*—Using machine to generate text has attracted considerable attention recently. However, low quality text generated by machine will seriously impact the user experience due to the poor readability. Traditional methods for detecting machine generated text heavily depend on hand-crafted features. While most deep learning methods for general text classification tend to model the semantic representation of topics, and thus overlook the semantic coherence that is also useful for detecting machine generated text. In this paper, we propose an end-to-end neural architecture that learns semantic coherence of text sequences. We conduct experiments on both Chinese and English datasets with more than two million articles containing manually written and machine generated ones. Results show that our method is effective and achieves the state-of-the-art performance.**

***Index Terms*—Deep Learning,Neural Network,Semantics,Text Classification**

I. Introduction

Machine generated texts are documents or sentences that are automatically written by following artificial rules or machine learning approaches. Recently, generating specific text has become a popular research topic and much work has made progress on it. It is a foundational task in many NLP applications, such as automatic text summarization [9], [10] and machine translation [3], [38]. In these tasks, texts are usually generated through automated processes [5] with supervised training such as the encoder-decoder model [6]. Despite the breakthroughs have been made, to put it into practical application still remains a big problem because there are no effective and automatic methods for checking the quality of the generated text. As a result, automatically generated documents, for example, news written by robot, like figure 1, have appeared on the web even without careful verification. This will hurt user experience and fool the web search engines because low quality documents are supposed to have low rank scores while they can rank high in search results due to web spamming [11]. Moreover, even some fake papers generated by machine have been used to manipulate the index score [20]. Therefore, there is a urgent need of an effective detection method to accurately identify the machine generated texts from the real ones.

Existing researches for detecting machine generated text are mainly based on a large range of carefully designed hand-crafted features which can evaluate the different properties of text samples, such as richness of vocabulary and POS [2],

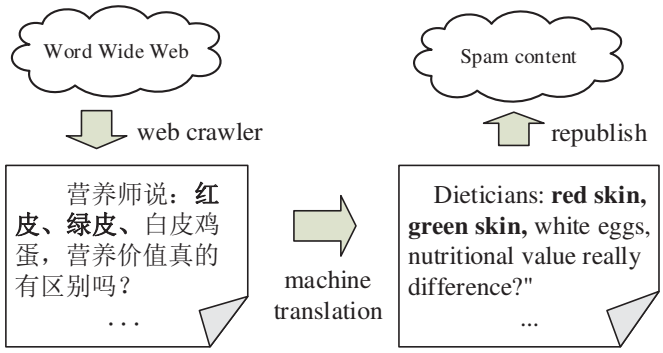


Fig. 1: Machine generated Spam text example: some web pages directly copy news texts from other languages source through machine translation and web crawler. "Red skin" which means "red eggs" is confusing caused directly poor translation.

[16]. For example, the richness of vocabulary is measured by the distinct ratio of words or n-grams. However, in order to generalize the methods on different data, especially to process corpus in different languages or generated by different algorithms, traditional approaches need to be exquisitely remodeled to fit the new dataset. In other words, different features need to be extracted and the model parameters need to be tuned, which brings substantial manual effort. On the other hand, deep learning methods have shown their superiority in text classification, which is quite similar to this task. General text classification aims to automatically extract text semantics [23] from the raw text. These methods include convolutional neural networks (CNN) [17], recurrent neural networks (RNN) [21] and RNN with attention mechanism [39]. However, most of them only intend to capture the semantic representation of texts so as to obtain the latent main topics or sentiments. Hence these models are insufficient for learning semantic coherence in machine generated text detection task. CNN-based models aim to capture the local features [17] directly over the embedding matrix via different sizes of convolution filters. A filter with n-size slide window can capture the n-gram feature [40]. But the word order information will be lost after the operation of max-over-time pooling. In fact, word order is one of the key features to measure the coherence of semantics. RNN-based models aim to capture the semantics of context. Long Short-Term Memory (LSTM), a variant of

978-1-7281-2009-6/$31.00 paper N-19674.pdf

RNN, is capable of learning long-term dependency and widely used in text classification. It usually tries to get the semantic representation through average or attention pooling applied to hidden states. This is effective in capturing the global meaning of a document, which is the key point in general text classification. However, such practices will also weaken the capability of models to obtain the semantic coherence.

In this paper, we propose a novel approach that concentrates on capturing the semantic coherence of texts to more accurately detect machine generated text. We refer to our model as SC-Net and formulate this task as a binary classification problem. In this task, semantic coherence represents the naturalness of the text. For instance, whether an article has word order errors or incompatible sentences. The basic idea is to exploit deep learning methods to make semantic errors (e.g., word order error) propagate as the length of text grows, such that the deep features of machine generated text is more discriminative. Specifically, we first train word embeddings only using the natural texts written by humans. Then, Bidirectional Gated Recurrent Units (Bi-GRU) in pre-trained Language Model(LM) are adopted to model the naturalness of both kinds of texts. At the same time, hidden units of Bi-GRU in pre-trained LM can fully obtain semantic coherence representations. Also, to reduce the influence of position where semantic incoherence occurs, we utilize max pooling on the obtained feature maps and combine the pooling results. Finally, the combined features are used for classification. Experimental results on different datasets indicate that our SC-Net outperforms existing state-of-the-art text classification approaches in detecting machine generated text.

Our contributions are as follows:

- We formulate the problem of detecting machine generated text when automatic text generation is rising to be a popular application in NLP community. Correspondingly, we provide two large scale datasets for further study.
- We propose a novel end-to-end approach that can automatically learn semantic coherence of texts instead of manually designing textual features.
- We evaluate the effectiveness of our approach against both traditional machine learning and deep learning based methods. Furthermore, we visualize the mechanism of capturing semantic coherence and do a theoretical analysis for better exploring this problem.

II. Methods

In this section, we first introduce the architecture of the proposed model SC-Net for machine generated text detection. Then, we explain the mechanism of modeling the semantic coherence and how it can extract such features automatically. The architecture of SC-Net is shown in figure 3. In our model, we make full use of GRU in pre-trained LM which have the memorizing capability [12] to model the semantic coherence. These semantic features are consecutive hidden states of LM that contain semantic incoherence locally. Thus we apply conventional filters to capture such local features for classification.

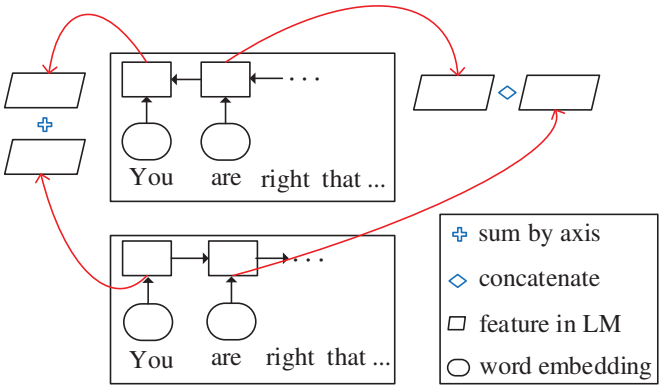


Fig. 2: Two ways to utilize features generated by bidirectional LM.

A. Pre-trained language model

Pre-trained word embedding shows great effect in many NLP tasks [28], [30]. But such static word embedding can not capture the polysemy. For Example, in the sentence "You are right that post office is on the right.", the word "right" is used as both an expression of directions and a definite attitude. However context semantics is essential to NLP systems. Recent work has also shown the top layer of LSTM can encoder context semantics [26]. ELMo [32], GPT [33] and BERT [8] show great gains from powerful context semantics. Similarly, we apply LM on unlabeled well written data for pre-trained to capture the task specific context semantics, that is encoding the semantic error and abnormal semantic. Different from above methods, we make use of all the time sequence features encoding by LM, rather than get a fix dimension sentence representation for a specific classification task. A Language model predicts a word sequence $s_1, s_2, s_3, ..., s_N$

$$p(s_1, s_2, s_3, ..., s_N) = \prod_{i=1}^{N} p(s_i | s_1, s_2, s_3, .., s_{i-1}) \quad (1)$$

In order to capture bidirectional feature in LM, a reverse LM is applied. A reverse LM predicts the previous word s_i by the future word sequence $s_{i+1}, s_{i+2}, ..., s_N$.

$$p(s_1, s_2, s_3, ..., s_N) = \prod_{i=1}^{N} p(s_i | s_{i+1}, s_{i+2}, .., s_N) \quad (2)$$

Both the LM and reverse LM can produce features by the hidden layer of LSTM or GRU. In our model, we use the LM and the reverse LM as two separate feature generator after removing the projection and softmax layer. We maximizes the log likelihood as following equation, θ_{gru} represents parameters in GRU while other parameters like softmax layer are θ_{other}:

$$\sum_{i=1}^{N} (\log p(s_i | s_1, s_2, s_3, .., s_{i-1}; \theta_{gru}, \theta_{other}) + \log p(s_i | s_{i+1}, s_{i+2}, .., s_N); \theta_{gru}, \theta_{other})) \quad (3)$$

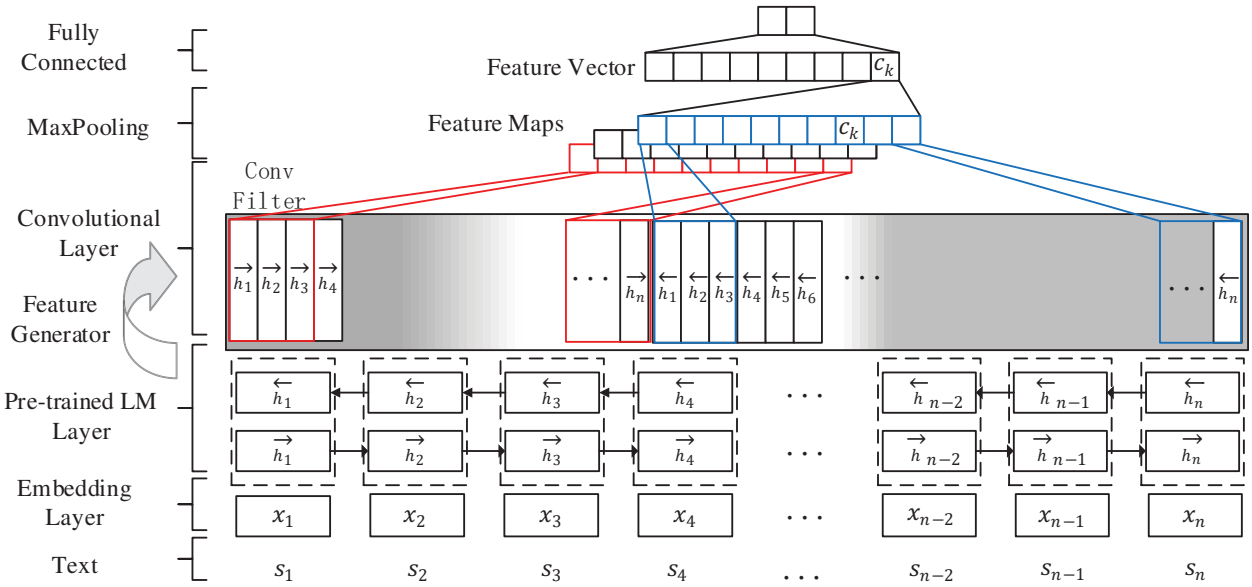


Fig. 3: The overall architecture of SC-Net.

More details about how to use GRU in pre-trained LM will discussed in the following subsection.

B. Semantic Coherence Modeling

In our task, we use GRU in pre-trained LM for modeling coherence. GRU in language model are able to propagate historical information in a time sequence, because it generates the current hidden state $h_i \in \mathbb{R}$ combing the current input $x_i \in \mathbb{R}$ with the previous hidden state h_{i-1}. The transition functions of GRU are as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\ r_t &= \sigma(W_t x_t + U_t h_{t-1}) \\ \tilde{h_t} &= tanh(W x_t + U(r_t \odot h_{t-1})) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \end{aligned} \tag{4}$$

Here $\odot$ stands for element-wise multiplication, σ is the logistic sigmoid function that outputs values in the range of $[-1, 1]$, and W_z,W_t,W,U_z,U_t,U adaptively select new vector and remove history vector for semantic composition. That is the mechanism of capturing long-term dependency in sequences. Since text can be regarded as a sequence of words, it is natively supported by pre-trained LM for modeling the semantic change. On top of this, we perceive that the bidirectional semantics is of equal importance, as the natural language usually has contextual information. Therefore, we apply the Bi-GRU to enhance the capability of LM for processing such sequential data.

We pre-train the word embeddings and store them as a look-up table. The input text sequence $s : [s_1, s_2, s_3, ..., s_n] \in \mathbb{R}$ will be transformed into the word embedding sequence $x : [x_1, x_2, x_3, ..., x_n] \in \mathbb{R}$. The n above is the length of text sequence.

The transformed words are high dimensional vectors and then fed into the Bi-GRU, core component in pre-trained LM, which can process the input vectors and generate one hidden state at each time step. For time step t, the input word s_t will be transformed into the embedding x_t. The hidden unit will generate the hidden state h_t. As mentioned above, we use pre-trained LM to model the semantics from both directions. The bidirectional LM has the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$ in Equation 5. The forward one process the embedding sequence form x_1 to x_n, and the backward one process the embedding sequence from x_n to x_1.

$$\begin{aligned} x_t^i &= W_e s_t^i, t \in [1, n] \\ \overrightarrow{h_t^i} &= \overrightarrow{LM(x_t^i)}, t \in [1, n] \\ \overleftarrow{h_t^i} &= \overleftarrow{LM(x_t^i)}, t \in [1, n] \end{aligned} \tag{5}$$

The $W_e \in \mathbb{R}^{d \times |V|}$ can be considered as the word embedding transform matrix or a look-up table. Here d is the dimension of word embedding and $|V|$ is vocabulary size.

After the pre-training procedure on well written data, we have two ways to combine the context feature in Figure 2, one is concatenating, another is summing by its axis. The summing way has a half parameters than concatenating. Two ways to make use of features generated by LM is formulated as

$$\begin{aligned} M &= [\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n}; \overleftarrow{h_1}, \overleftarrow{h_2}..., \overleftarrow{h_n}]. \\ M' &= [\overrightarrow{h_1} + \overleftarrow{h_1}, \overrightarrow{h_2} + \overleftarrow{h_2}, ..., \overrightarrow{h_n} + \overleftarrow{h_n}]. \end{aligned} \tag{6}$$

Then we obtain the matrix M, M' formed by the features of all time steps, which can represent the semantic consistency and continuity of the input sequence. The reason is that when there is a semantic incoherence at time t, such information

will be captured by both $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$. More importantly, it will also be transmitted along time steps in both directions, that is, LM propagates this feature to make the incoherent time longer. In this way, we construct semantic coherence features automatically. The mechanism of feature extraction will be discussed in the following subsection.

C. Semantic Coherence Feature Extraction

CNNs are proved to be not only superior in computer vision, but also effective in natural language processing. In our model, we treat CNN as an automatic feature extractor whose input is the semantic coherence matrix. Note that semantic error in a given text can be tiny, for example, only two words are out of order. Therefore, we apply a global max pooling layer following the convolution filters to produce the most distinctive feature vectors. As explained above, $h_t^{(i)} \in \mathbb{R}^m$ represent the semantics at time t, and also denote t-th row in semantic matrix $M^{(i)}$. A convolution filter m convolves at each position to generate a feature map c as like as processing an image by Equation 7:

$$c_j = f(W_c \odot h_{i:i+k-1}^{(j)} + b), \tag{7}$$

where $b \in \mathbb{R}$ is a bias term and f is the nonlinear function such as rectified linear unit($ReLU$), $tanh$ and $sigmoid$. In our case, we choose $ReLU$,defined as $f(x) = max\{0, x\}, x \in \mathbb{R}$, as the nonlinear function. After convolution operation on the semantic matrix $M^{(i)}$, we get the feature maps. Each convolution filter generates a feature map $c \in R^{l-k+1}$, where l and k are the sequence length and the kernel size of a filter respectively. Global max pooling is applied to the feature map to get the most important feature. Given a feature map $c \in \mathbb{R}^p$ of length p, global max pooling choose the highest value in c.

The semantic coherence features are the outputs of global max pooling on feature maps of both directions, denoted as $c^{(f)}, c^{(b)} \in \mathbb{R}^{n \times p}$. We can concatenate them as follows:

$$V = [c_1^{(f)}; c_2^{(f)}; ...; c_n^{(f)}] \oplus [c_1^{(b)}; c_2^{(b)}; ...; c_n^{(b)}]. \tag{8}$$

Here n is the number of filters and $\oplus$ is the concatenation. Thus we get the final abstract feature vectors $V \in \mathbb{R}^{2n}$ and use it as the input of fully connected layer for determining whether a text is generated by machine.

D. Training

We conduct the supervised learning for training the model. We denote the set of training data as $\mathcal{X}$ and the label of data as $\mathcal{Y}$. In our case, machine generated text detection is a binary classification task with negative and positive labels. The parameters of model to be trained are defined as Θ. For each $x \in \mathcal{X}$, the model gives the probability $p(y; x, \Theta) \in [0, 1]$ of positive label y. We choose the sigmoid function to obtain such probability before computing the binary cross-entropy loss $\mathcal{L}(\Theta)$. The target of training is to optimize the binary cross-entropy as follows:

$$\mathcal{L}(\Theta) = -\frac{1}{\mathcal{N}} \sum_{x \in \mathcal{X}} [\widetilde{y} ln p(x) + (1 - \widetilde{y}) ln(1 - p(x))], \tag{9}$$

where $\widetilde{y}$ is the ground truth of sample x and $\mathcal{N}$ is the total number of samples in dataset$\mathcal{X}$. We use the adamax [18] to optimize the loss function. Adamax is a variant of adam [18] based on the infinity norm, an algorithm for first-order gradient-based optimization of stochastic objective functions with adaptive estimates of lower-order moments. For regularization, we apply the dropout [34] before the fully connected layer. Finally, if $p(x)$ is less than the threshold θ, we deem x as machine generated text.

III. Experiments

We conduct experiments to evaluate our approach to machine generated text detection. We describe datasets, experimental setting and methods to compare in this section.

A. Dataset Description

We test all the methods on two different datasets including Chinese and English corpus respectively. We will present information about each dataset in the following paragraph and summarize the statistics in Table I.

There are three main ways of generating text, including using artificial rules, Markov model and RNN-based model. However, those generated texts are often incomprehensible due to various kinds of semantic errors. For example, a generated sentence can be very long or have many repetitive words etc. Among all the errors, word order error and incompatible words/sentences are most representative and difficult to recognize. For instance, "The new economy *looks to forward* the collaborative policy" and "The new economy looks forward to the collaborative policy *after they got the salary*" should be "The new economy looks forward to the collaborative policy". The semantic errors are in italics.

To this end, we provide such datasets that mainly include the above errors. Initially, we collected a dataset presented by Qihoo 360 Search that contains over one million samples whose contents are Chinese daily news. Apart from normal news that are deemed as positive samples, there are also machine generated news serving as negative samples including machine translation tools generated, text auto summarization generated and machine writing-bots generated articles. The typical features of negative samples include word order error and sentence whose meaning is incompatible with the topic of a document. There are also some repetitive words or sentences in a few negative samples. For further evaluation, we downloaded the New York Times(NYT) corpus. NYT corpus contains nearly every article published in the New York Times between January 01, 1987 and June 19th, 2007. As a large scale corpus, NYT was widely used in document routing, document categorization, entity extraction, cross document coreference resolution, and information retrieval etc. We randomly converted selected NYT articles into negative samples by making small perturbations(e.g, swapping positions of two words) to the words or sentences. Another part of negative samples are generated by translating NYT articles into Chinese and back to English. The other part generated by text auto summarization. In this way, we produced a new dataset of

English written news that is analogous with the Chinese dataset.

For illustration purposes, we denote the two datasets as QiHoo Generated News(QHGN) and Modified New York Times(MNYT). Both datasets do not come with separate test sets since they are reprocessed, thus we randomly divided all samples into 90% for training set and 10% for test set.

TABLE I: Summary of datasets:positive, negative samples number and average, max token number.

Dataset	Positive	Negative	Avg token	Max token
QHGN	420,632	599,531	605	2,618
MNYT	599,994	417,521	1,306	69,517

B. Experimental Settings

We conduct experiments on the datasets mentioned above. We train all the model with an early-stopping strategy and save the best parameters. To speed up the training process, we train our model on GPUs which benefits from parallel computation of the tensors. For data processing, we tokenize Chinese corpus using **jieba**, and tokenize and lowercase English corpus using **NLTK**. Word embeddings trained by **word2vec** are used in our model. LM is pre-trained on the large 400 millions English Wikipedia corpus and 400 millions Chinese Wikipedia corpus. The length of document in evaluating task is restricted to 1000 and the exceeding words are truncated. In our model, we set the number of GRU cells in LM to be 400, the filter size to be 3, the dimension of word embedding to be 256 and the dropout rate to be 0.2. For classification, we set the threshold θ that mentioned in Section 2 to be 0.5. The evaluation metrics are $F1$-score $F1 = \frac{2*PR}{P+R}$ and $accuracy$.

C. Methods for Comparison

We compare our method with several state-of-the-art approaches in general text classification:

- **SVM** We choose SVM, a classical statistical machine learning method, using handcrafted features such as tf-idf, n-gram, word count, POS and word richness etc. Word richness can be measured by the ratio of vocabulary size and total number of words.
- **TextCNN** [17] This method adopts convolution operation on the word embedding to get the text representation. Multi-channel CNNs are used so that different sizes of kernels can capture the n-gram features.
- **LSTM** The hidden state vectors of LSTM are averaged or applied by attention mechanism to get the text representation.
- **RCNN** [21] This method learns representation that combines each word in the sentence or document with left side context and right side context.
- **HAN** [39] This method has a hierarchical structure with attention mechanisms applied at the word-level and sentence-level, enabling it to capture important features of both levels.
- **fastText** [13] Word representations are averaged for obtaining the text representation. In our experiments, n-gram features are used for classification.
- **Structured** [24] This model achieves state-of-the-art results on document modeling tasks. Structure-aware document representations are learned from data without recourse to a discourse parser or additional annotations. This model encodes a document while automatically inducing rich structural dependencies.

Among them, there are mainly two categories: statistical machine learning methods using hand-crafted features and deep learning methods using neural networks. These neural networks can be further categorized into CNN-based, RNN-based and mixed methods. For example, RCNN takes advantages of both CNN and RNN.

IV. Results and Analysis

A. Discussion of Results

The results on QHGN and MNYT datasets are shown in table II. On the one hand, we can see that our model significantly outperforms traditional methods, i.e., SVM with hand-crafted features. On the other, our proposed method also achieves best results on both datasets. This demonstrates that our method indeed learns semantic coherence features while the general text classification methods pay more attention to the semantics itself.

Comparing SVM with deep learning methods on QHGN, we can see that deep learning methods using automatic feature extraction exceed SVM using hand-crafted features by more than 19%. Also, fastText which is the simplest neural model using n-gram features still performs well. Results of CNN-based and RNN-based models indicate that RNN-based models are more suitable for this task. LSTM with attention performs better than LSTM with average pooling, and both of them are comparable to RCNN. It proves that attention mechanism can boost the weight of particular hidden state which contains more important features. Structured and HAN are both hierarchical models, but Structured achieves much better result than HAN on QHGN. This can be explained that Structured can capture the structural information of text which is more important than just semantics. However, this model is tricky in that it can not coverage on MNYT although we preprocess the MNYT corpus for it the same as for HAN. We conjecture that Structured is sensitive with the length of sentences and documents because MNYT contains many short sentences and a document can be very long(see Table II). Meanwhile, it consumes larger GPU memory and spends much more time in training than all other models, which leads to being difficult for fine-tuning.

The performance of TextCNN is worse than RNN-based model on QHGN. Features generated by convolution filters using max-overtime-pooling will lose the word order information, which is central in modeling the semantic coherence. RNNs can encode the semantics of context into hidden states and learn long term dependencies better. But most of the

TABLE II: Comparison of results on QHGN and MNYT dataset.

Models	QHGN		MNYT	
	F_1 score	Acc	F_1 score	Acc
SVM	0.4230	0.5816	0.7159	0.6090
TextCNN [17]	0.8489	0.8619	0.8122	0.7553
fastText [13]	0.7248	0.7757	0.7051	0.6412
LSTM+Attention	0.9006	0.9175	0.8076	0.7424
LSTM+Average	0.8870	0.9064	0.7858	0.6982
RCNN [21]	0.8936	0.9116	0.8786	0.8451
HAN [39]	0.8474	0.8677	0.8307	0.7667
Structured [24]	0.9146	0.9268	-	-
SC-Net(non-static pre-trained LM)	**0.9234**	**0.9364**	0.8935	0.8682
SC-Net(random initialized parameters without LM)	0.8823	0.9013	0.8875	0.8552
SC-Net(static pre-trained LM)	0.9219	0.9343	**0.9034**	**0.8783**

baseline models do an operation of combining the semantics representation via averaging or attention pooling, thus the local semantic error is more likely to be ignored. While on MNYT, TextCNN performs better than LSTM models. This can be explained that local features are more effective than contextual features when the text is too long.

TABLE III: Comparison of results by different merge mode applying to hidden states by bidirectional GRU in LM.

Merge Mode	QHGN		MNYT	
	F_1 score	Acc	F_1 score	Acc
sum by axis	0.9173	0.9251	0.8916	0.8659
concatenate	0.9219	0.9343	0.9034	0.8783

TABLE IV: Comparison of results by different size parameters in Language models. All language models use same parameters except single or bidirectional, numbers of hidden states and projections.

Language Model	QHGN		MNYT	
	F_1 score	Acc	F_1 score	Acc
Single GRU-256-256	0.9156	0.9279	0.8912	0.8631
Single GRU-400-256	0.9178	0.9283	0.8923	0.8643
Bidirectional GRU-256-256	0.9189	0.9308	0.8997	0.8705
Bidirectional GRU-400-256	0.9239	0.9343	0.9034	0.8783
Bidirectional GRU-1000-256	0.9193	0.9312	0.9013	0.8694

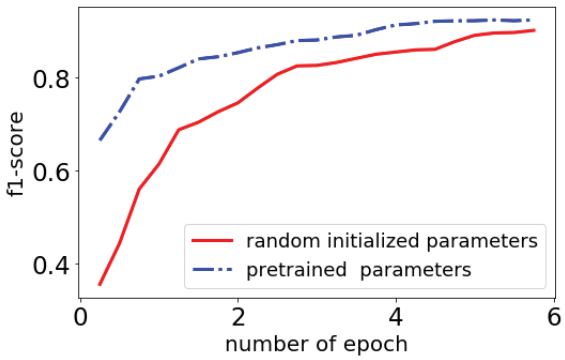


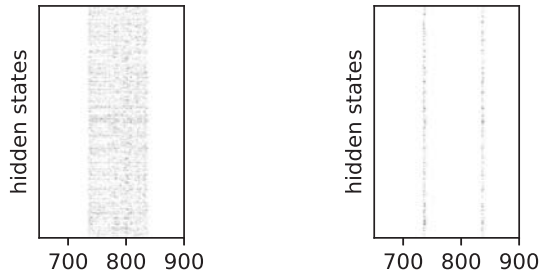
Fig. 4: Comparison of results by random initialized parameters in GRU and the pre-trained one.

Further experimental results and observations are as follows:

- Pre-trained LM shows great improvement in this task. This suggests that unsupervised pre-trained LM contains a lot of priori knowledge that is advantageous to semantic representation. In contrast, supervised model structure, that is, SC-Net, random initialized parameters without Language Model, is prone to be overfitting. What's more, pre-trained parameters in LM reach a faster convergence result as shown in Figure 4. After initialized with pre-trained parameters in LM, it exceeds the random initialized one's best result at epoch 3.
- Hierarchical models like HAN and Structured take more time to converge because they encode both sentences and documents at the same time. To realize such approaches, they set up the max length of a sentence and the max number of sentences in a document. However, it suffers from the variation of the sentence length, for instance, sentences and documents must be processed to be fixed length, which breaks the semantic coherence or even clips wrong sentences.
- Traditional text classification methods usually remove stop-words and punctuations, for these tokens have no useful information about semantics. But in this task, stop-words and punctuations are indispensable, because the position extremely influences the grammar correctness. We have tried the Google pre-trained embedding, which has no punctuations and degradation in performance is over 10%.
- Comparison of results by different merge mode applying to hidden states by bidirectional LM are shown in Table III. Using concatenate to build the semantics coherence matrix can give us more information generated by bidirectional LM.
- Comparison of results by different parameters in LM are shown in Table IV. We can learn that bidirectional LM can help a lot in capture context features. We decreased model size to 400, for 1000 hidden states show more time consumption and less accuracy promotion.

B. Visualization and Theoretical Analysis

Now we dig into the mechanism of modeling semantic coherence. We visualize the matrix of hidden states in the process of inference. Thus the difference of parameters between negative and positive samples will be shown in the visualization. In another words, if we treat the matrix of hidden states as an image, the incoherent area will show the difference from normal coherent sequence. As shown in Figure 5, we take



(a) Replace a sentence with one from another document.

(b) Change the position of two words.

Fig. 5: Here is the visualization to explain the mechanism of capturing semantic coherence. As shown in this figure, inserting sentence will take a large area of semantic incoherence in a normal sentence. Changing the position of two words will also cause semantic incoherence around the position of exchanged words.

out the hidden state matrix of normal coherent text sequence and minus the matrix of negative samples. The result proves that our model captures the incoherent features so that it can distinguish positive samples from negative samples effectively. As shown in Figure 5(b), if we exchange the position of two words in text sequence, for example, "*Deep* learning is a state-of-the-art tool in natural language *processing*" to "*processing* learning is a state-of-the-art tool in natural language *Deep*", there are two obvious incoherent intervals where the words are exchanged. Considering such local features, we believe that convolution filters can take full advantage of them.

It is worth mentioning that the incoherent area is an interval rather than just a point in space, which is also shown in Figure 5. The memory mechanism of RNNs may explain it, that is, RNNs can store long-term dependencies. Therefore the incoherent information will spreads along with the text sequence but will not last too long. Next we attempt to explore how long it could propagate that caused by the change of input x_t.

If we set U_z,U_t,U to zero matrix, then we get the simplified GRU as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t) \\ \tilde{h_t} &= tanh(W x_t) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \end{aligned} \tag{10}$$

Suppose there is a change $\overrightarrow{\Delta x}$ of x_t at time t, then we get $x'_t = x_t + \overrightarrow{\Delta x}$. So there is a change on hidden state $\Delta h_t = \Delta z_t \odot h_{t-1} + \Delta z_t \odot \tilde{h_t} + (z_t + \Delta z_t) \odot \Delta \tilde{h_t}$, where Δz_t and $\Delta \tilde{h_t}$ are the change of two gates, and Δh_t denote the change of hidden state at time t. This change will spread by $h_{t+i} = (1 - z_{t+i}) \odot h_{t+i-1} + z_{t+i} \odot \tilde{h_{t+i}}, i \in N^+$. We denote ϵ_t as one element in Δh_t. As shown in Fig 5, ϵ_t propagating with sequence causes the shadow area, whose gray shade measures the scale of semantic change. We assume $1 - z_t = p$, $p < 1$ and $z_{t+i} \odot \tilde{h_{t+i}} = q$, thus $\epsilon_{t+i} = \epsilon_t p^{i-1}$. In another words, it decays p at each time step, ϵ_{t+i} will vanish with iteration. How long ϵ_{t+i} will converge to zero is determined by p. Actually, p changes with the iteration, for forget gates can learn how much to forget automatically and adaptively. We can see in Fig 5, ϵ_t propagation will last a few time steps in the case of exchanging two words.

V. Related Work

The existing work of detecting machine generated text can be classified into two categories. One mainly bases on statistical machine learning method using hand-crafted features. Lavoie [22] propose several features like word repetition score and reference score to detect machine generated academic papers. Tien and Labbé [36].Amancio [1] present a complex network(CN) to recognize SCIgen generated documents. Nguyen and Labbé [27] use distance/similarity measurement to detect automatic generated text. All methods above need hand-crafted features on different data.

Recently, motivated by the success of deep learning in many domains, several deep learning methods for text classification have been presented. Kalchbrenner et al. [14] used a dynamic CNN for sentence modeling, and show significant improvements over traditional tasks such as sentiment classification. Another novel model proposed by Kim [17] applied a much simpler CNN using different kinds of word embedding to sentence classification and get competitive results. Graph CNN has also been used to handle large scale text classification [29]. Except for CNN-based models, RNN-based models also have excellent performance. For example, hierarchal RNN has been proposed for long document classification [35] and later attention mechanism [39] is also introduced to emphasize important words and sentences. Recently Liu and Lapata [24] proposed a method that can automatically induces rich structural dependencies. However, all of these models aim to model the semantics into a single vector.

Language models are important in many NLP systems, such as statistical machine translation [19]. Language models [4] in neural network methods are also applied in machine translation. LMs above are usually single forward. There are also bidirectional LMs used in machine translation [31] and handwriting recognition [25]. Recently, how to interpret RNN states have been attracted some researchers. Karpathy et al. [15] use character-level language models to reveal the existence of interpretable cells that keep track of long-range dependencies such as line lengths, quotes and brackets. Our work using GRU in LM shows that RNN can encode interpretable features to capture the semantic coherence. Semantic coherence modeling using CNNs [7], [37] aimed to get a coherence score or sentence ordering between several sentences. Here we capture the semantic coherence features for machine generated text detection and evaluate these models on both Chinese and English datasets.

VI. Conclusion

We have proposed a novel method that learns the coherence of semantics for machine generated text detection. Gated recurrent neural networks in Language Model are effective to

model the semantic coherence. Convolution filters are applied to extract the features of semantic coherence from the hidden state matrix generated by Language Model. We evaluate our approach on machine text detection task with different datasets and experimental results show its advantages. The mechanism of capturing semantic coherence is further explored by both visualization and theoretical analysis.

VII. Acknowledgments

This work is supported by NSFC program (No. 61872022, 61421003), SKLSDE-2018ZX-16 and partly by the Beijing Advanced Innovation Center for Big Data and Brain Computing.

References

[1] Diego Raphael Amancio. Comparing the topological properties of real and artificially generated scientific manuscripts. *Scientometrics*, 105(3):1763–1779, 2015.

[2] Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2), 2004.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] Daria Beresneva. Computer-generated text detection using machine learning: A systematic review. In *International Conference on Applications of Natural Language to Information Systems*, pages 421–426. Springer, 2016.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Baiyun Cui, Yingming Li, Yaqing Zhang, and Zhongfei Zhang. Text coherence analysis based on deep neural network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2027–2030, New York, NY, USA, 2017. ACM.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Mohamed Abdel Fattah and Fuji Ren. Automatic text summarization. *World Academy of Science, Engineering and Technology*, 37:2008, 2008.

[10] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

[11] Zoltan Gyongyi and Hector Garcia-Molina. Web spam taxonomy. In *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[15] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[16] Martin Kay and Christian Boitet, editors. *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. Indian Institute of Technology Bombay, 2012.

[17] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[19] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[20] Cyril Labbé. *Ike Antkare one of the great stars in the scientific firmament*. PhD thesis, LIG, 2010.

[21] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[22] Allen Lavoie and Mukkai Krishnamoorthy. Algorithmic detection of computer generated text. *arXiv preprint arXiv:1008.0706*, 2010.

[23] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, 2017.

[24] Yang Liu and Mirella Lapata. Learning structured text representations. *arXiv preprint arXiv:1705.09207*, 2017.

[25] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888, 2015.

[26] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.

[27] Minh Tien Nguyen and Cyril Labbé. Engineering a tool to detect automatically generated papers. In *BIR 2016 Bibliometric-enhanced Information Retrieval*, 2016.

[28] Hao Peng, Mengjiao Bao, Jianxin Li, Md Zakirul Alam Bhuiyan, Yaopeng Liu, Yu He, and Erica Yang. Incremental term representation learning for social network analysis. *Future Generation Computer Systems*, 86:1503–1512, 2018.

[29] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1063–1072. International World Wide Web Conferences Steering Committee, 2018.

[30] Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. Incrementally learning the hierarchical softmax function for neural language models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[31] Alvaro Peris and Francisco Casacuberta Nolla. A bidirectional recurrent neural language model for machine translation. 2015.

[32] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*, 2018.

[34] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[35] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432, 2015.

[36] Nguyen Minh Tien and Cyril Labbé. Curious cases of automatically generated text and detecting probabilistic context free grammar sentences with grammatical structure similarity. In *39th European Conference on Information Retrieval*, 2017.

[37] Liang Wang, Sujian Li, Yajuan Lv, and WANG Houfeng. Learning to rank semantic coherence for topic segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1340–1344, 2017.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489, 2016.

[40] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.