LaTeX Author Guidelines for CVPR Proceedings

First Author
Institution1
Institution1 address
firstauthor@i1.org

Second Author
Institution2
First line of institution2 address
secondauthor@i2.org

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous CVPR abstracts to get a feel for style and length.

1. Introduction

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sticky tape to attach your artwork to the paper), so all authors should read this new version.

2. Current methods

In recent years, the field of Natural Language Generation (NLG) has expanded its repertoire of solutions for text generation to include a variety of methods. Models such as RNN's, Bi-LSTM [*], Transformers-based models (e.g. GPT [*]), GAN's, among others, have evolved to produce elaborate pieces of text for different specifics tasks. Each method proposes a different approach to the task at hand.

2.1. RNN

Recurrent Neural Networks (RNN) are commonly used in sequence-to-sequence (seq2seq) tasks, such as text generation. Simple RNN's have been adapted to overcome its own limitations, e.g. vanishing or exploding gradients, introducing concepts such as Long Short Term Memory networks (LSTM) or even bidirectional LSTM's (Bi-LSTM). Bi-LSTM have been used in many cases for text generation [*] improving its performance in some cases [*]. These models are still used in recent years for text generation in different languages [*]. Some studies [*] use external word embeddings for the training process which can be seen as pre-trained models for text generation.

In this work, we train a simple RNN with an Embedding layer, one LSTM module and a dense output layer. The task of the RNN is to predict the next word for the text generation. For this net, several hyperparameters can be set such as the number of LSTM layers, whether those layers are bidirectional, pre-trained word embeddings, among many others. The configuration that best minimized the cross-entropy loss was an embedding layer with no pre-trained embeddings, three bidirectional LSTM layers and one dense output layer. With this configuration, 1000 sentences were generated for comparison with alternative methods.

2.2. Transformers

With the introduction of [*], new models for text generations task have arisen. [*] introduces an Encoder-Decoder based architecture for machine translation tasks. This architecture was ingeniously modified by [*] and [*] to yield models such as BERT and GPT. Both Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT) are based on the idea of transformers and can be finetunned and adapted for a variety of tasks [***]. These models have slowly replaced the RNN approaches. As [*ModernMethodsOfTextGeneration] states: "Transformers disrupted sequence-based deep learning significantly. The two variants of transformer models that we showed, BERT and GPT-2, outperform previous approaches with RNNs".

GPT is widely known as a powerful tool for text generation. Conversely, the BERT model alone has not been accepted as a strong generative model. Some authors [**?**] have proposed workarounds to BERT limitations for text generation.

3. BertTextGenerator

BERT is a language representation model designed to achieve state-of-art results in several natural language processing tasks such as question answering and language in-

ference. BERT is trained on masked language modeling objective, in which it predicts a word given its left and right context. Given this behavior, it is possible to deduct that BERT is not the most indicated model for text generation tasks. However, [**?**] have demonstrated why it is possible to use BERT as a traditional language model by means of showing that BERT is a combination of a 'Markov random field language model (MRF-LM), with pseudo Log-Likelihood training'.

This conclusion allows using BERT as a generative model of sentences to either score a sentence or sample a sentence. To score a sentence with BERT is possible to compute the unnormalized log-probabilities of a set of sentences and then, based on this, it is possible to sort the set of sentences according to their probabilities. To sample a sentence with BERT it is necessary to start with a random initial state which is initialized to be an all-mask sequence, i.e. ([MASK], ...,[MASK]). Then, at each iteration i, a position is selected in a random uniform way and mask out to be replaced by a new token according to the probability of the token being present in the current sequence of tokens.

Several experiments have been made to demonstrate the potential of BERT as a text generation algorithm. With the obtained results was possible to conclude that even when the BERT generations are of the worst quality, compared with models like GPT, the text generated by BERT is more diverse and this is a meaningful result.

3.1. Attention mechanism

The attention mechanism is composed by a feed forward neural network that is trained to identify the more relevant components in the context in order to predict a new token.

For example, take the sentence "the animal didn't cross the street because it was too tired", as a human is easy to know that in the sentence the pronoun "it" refers to the subject "the animal" and the verb "was" is the correct conjugation of the verb "To be" for the third person, but for an algorithm is not easy to know these relationships and that is the task of the attention mechanism.

The FIGURE shows how the attention mechanism create a relationship between the token "it" and several components of the sentence. For example, the first attention head is totally focused on the verb "was", the second and third attention heads are focused on the special tokens "[CLS]" and "[SEP]" that represent the beginning and the end of the sentence respectively, finally the fourth attention head is focused on the word "animal". In this way BERT can understand which parts of the context are more relevant for each one of the tokens and when a token is masked to be replaced on which parts of the context must focused to make the prediction of the new token.

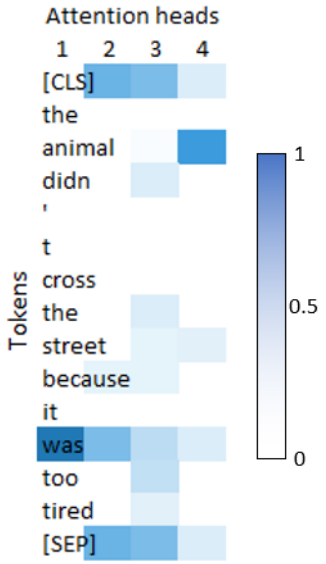

Figure 1. Missing values per column in training dataset.

3.1.1 Using the attention mechanism to improve the sampling method.

When a token is predicted, it is possible to know which tokens of the context were more relevant to retrieve the prediction based on the attention weights. This information is a very meaningful insight to improve the way in which the sampling method is made.

We propose a new sampling method based on the attention weights returned by BERT when a prediction is done, the aim of this new sampling method is to force the model to choose the token that is going to be mask in a smarter way, below we describe the procedure to develop the sampling method in the iteration t:

- Step 1: Retrieve the attention masks of the iteration $t-1$ (a), then get the average value of the attention masks $A_{p,e}$ for each position p inside each one of the encoders e.
- Step 2: Get the average value of the attention mask for each position over all the encoders $totalA_p$.
- Step 3: For each position p, there is a counter c that register how many times the position p has been selected to be replaced.
- Step 4: The average value of the attention masks $totalA_p$ is divided by the counter c to create a penalization.
- Step 5: Normalize the result of the step 4 to get a probability distribution P.
- Step 6: Sample randomly with distribution P the new token to be replaced.

In the step 1 we summarize the information of the attention inside each one of the encoders, then in the step 2 we combine the information of the different encoders to obtain a single measure of the attention, the step 4 is useful to help the method to avoid that the sampling method get stuck sampling always the same positions given its strong relation with the current context, finally the steps 5 and 6 creates a probability distribution based on the penalized attention and sample the new token to be replaced.

3.2. Finetuning

BERT is a powerful model pre-trained on a broad corpus composed by the whole Wikipedia and Brown corpora. Typical fine-tuning techniques depends on the final task that the user wants BERT to perform. For example, a classical application of the model is to predict the sentiment of a sentence. In this case the focus of the fine-tuning would be on the [CLS] token, a special token specifically used for classification tasks.

In this case however the task is different and unusual for BERT. Since we need to use BERT to generate text, it is important that during the fine-tuning the model understands the structure of the text.

We have implemented a fine-tuning method that gave complete freedom to the user to decide the language, the structure and even the sentiment of the text to generate. The fine-tuning is performed considering only one task of the two original used to pre-train BERT; that is the mask-prediction. 15% of the tokens of each sentence are replaced with a masked token and BERT have to predict the original tokens. The loss is computed as the cross-entropy between the logits for the masked tokens outputted by BERT and the original tokens. This method allows the final user to start from a pretrained model or, if it is not available for the language chose by the user, from a cross-language model and fine-tuning on a specific text corpus.

The fine-tuning allows also to exploit the enormous potentialities of BERT tokenizer. The default vocabulary of the tokenizer contains 1000 unused tokens, whose weights are randomly initialized. Typically, these tokens are replaced with domain specific words, so that during the fine-tuning the model will be able to learn them. We have extended this idea in order to comprehend also tokens that defines the structure of the text like '\n' '\t'. Note that replacing in the vocabulary, for example, the token '[unused1]' with '\n' would have no effect since the tokenizer would remove these tokens from the text even before the tokenization, during the normalization step. To solve this problem we have implemented a Formatter class that helps the user by maintaining a map between some user specified tokens that needs to be preserved and some unused tokens chose to replace them. The user-specified tokens are replaced with the unused tokens before the tokenization and the fine-tuning and are replaced back only after the text generation. Using this method we were able to make a model learn the tercet structure of Dante's Divine Comedy.

3.2.1 Sentiment generation

BERT tokenizer gave also the possibility to define some special tokens. We have taken advantage of this to define a new method to generate text with a specific sentiment. Considering a set of sentences each one with a possible sentiment labels in the set ['pos', 'neg'], we define n (3 by default) new special tokens for each possible sentiment. The special tokens are of the type [pos-i] $i = 1, ldots, n$. Before the fine-tuning, the special tokens corresponding to the sentence sentiment label are appended at the beginning of the sentence. In this manner, the model, during the fine-tuning, is able to build a relation among the special tokens of a sentiment and the specific words related to that sentiment.

At inference time, to generate some text with a specific sentiment, we simply pass the special tokens of that sentiment as seed_text to the generate method.

This method, even in its simplicity, showed its efficacy in generating positive and negative italian tweets about football.

4. Experiments and evaluation

5. Conclusions