Neural Text Generation

Francesca Avidano
s291439

Gioana Teora
s267379

Tommaso Toso
s267448

Abstract

In this report, we make a comparison between text generation results obtained using the sampling procedures presented in [14] exploiting BERT ([6]), and the ones returned by traditional language models exploiting history to predict the next word, such as OpenAI GPT ([11]) and Transformer-XL (TFXL [4]). Specifically, we apply the Texygen metrics ([17]) over WikiText-103 in order to confront the generated texts that we got from all the aforementioned models.

1. Introduction

In recent years, researchers in the field of text generation have turned their attention to neural networks. Deep learning methods achieved great empirical success on a variety of neural language tasks, included text generation. In this report, we present the novel approach to the text generation task proposed in ([14]). The authors implemented a procedure to generate text from BERT ([6]) pre-trained on a *masked language modeling (MLM)* objective. Instead of predicting the next word in a sequence given the history, masked language models predict a word given its left and right context. Although it is not immediate to understand how models of this type can be exploited for neural text generation, the authors of [14] showed that this can be done by resorting to a sampling procedure based on Gibbs sampling.

For comparison purpose, we took into consideration two other models: OpenAI GPT ([11]) and Transformer-XL ([4]). They are transformer-based models and, at the time they were published, they both achieved state-of-the-art result on the task that they aim to tackle. GPT is a task-agnostic system combining transformers ([13]) and unsupervised pre-training. Transformer-XL, instead, contains a recurrence mechanism used to maintain temporal coherence while learning new token dependencies.

The measures exploited to carry out the comparison were taken from the Texygen bench-marking platform ([17]). In the evaluation phase, we took into consideration both the quality and the diversity of the generated text, resorting to several of the available measures and we took WikiText-103 as reference text. We also computed perplexity for all models. It turns out that, despite left-to-right language models return portions of text of higher quality, sentences obtained from BERT reach good results in terms of diversity.

Finally, we decided to exploit the sampling technique presented in [14] to implement an interactive platform for supporting Italian Twitter users when they want to type a tweet in Italian and then get its English translation. You can find it in the appendix A.

2. BERT for Text Generation

The first text generation model we present is the one proposed in [14]. It employs BERT (Bidirectional Encoder Representations from Transformers) ([6]) pre-trained on a MLM objective. Unlike a traditional language modeling objective, where predictions on the next word are based on the previous ones, masked language modeling predicts a word given its left and right context.

In order to train a deep bidirectional representation, a percentage of the input tokens is masked at random, and then predictions on those masked tokens are performed resorting the to cross entropy loss function.

We use WordPiece embeddings, a subword tokenization algorithm ([15]), with a 30,000 token vocabulary. The LM masking is applied after WordPiece tokenization with no special consideration given to partial word pieces.

We took a pre-trained BERT model trained on a mix of Toronto BooksCorpus (800M words) (TBC, [16]) and English Wikipedia (2,500M words) and its PyTorch implementation provided by *HuggingFace*.

2.1. Sampling strategy

Even if BERT is not a Markov Random Field, as stated here, it is a generative model and it can speak.

Let us consider a sentence with T tokens

$$X = (X_1, \dots, X_T),$$

where X_t are categorical random variables over the vocabulary used to pre-train BERT and the vector $X_{\setminus t}$ represents vector X after removing the t-th component. In

[14], the authors suggest to use a Gibbs Sampler to perform text generation. Specifically, given the conditional distributions $p\left(X_t|X_{\backslash t}\right)$ output by BERT, after initializing X as $X^0 = ([MASK], \ldots, [MASK])$ the algorithm will enter a loop and at each iteration i:

1. an index t^i will be randomly chosen from $\{1, \ldots, T\}$;
2. X_{t^i} will be masked out and a token $x_{t^i}^i$ will be sampled according to probability $p(X_{t^i}|X_{\backslash t^i} = x_{\backslash t^i}^{i-1})$ following one of these rules:
    - sampling only from the top k probable words (not allowed in burnin period);
    - sampling from all distributions;
    - choosing the most probable word.

In the following, we will refer to this sampling technique as **Parallel-Sequential** method.

Notice that we decided to start from the all $[MASK]$ vector, although it is possible to use any random sentence as initial condition. Furthermore, the first token of every sequence is always a special classification token ($[CLS]$), while the $[SEP]$ token is used to separate sentences.

In addition to this, we decided also to resort to the temperature parameter. This parameter allows to calibrate the confidence level of the model on the words to be chosen by affecting their probability of occurrence.

2.1.1 Sequential and parallel sampling strategies

In [14], two additional sampling methods are provided:

- **Sequential**. In this setting, at each time step t, we mask out X_{t^t}, we generate a word for that position and we plug it into the sequence. In this case, we do not give to BERT the entire sequence as input. We just give to it $\left(x_1^1, \ldots, x_{t-1}^{t-1}, [MASK], x_{t+1}^0, \ldots, x_{t+\text{leed\_out}}^0\right)$, where leed_out $< T$ is a user-specified parameter. After T timesteps, we have generated a word for all positions and we can terminate the process or repeat it from the current sentence. With this strategy, we sample from left to right, like the majority of text generation models.
- **Parallel**. In this case, we start with an initial sequence of all $[MASK]$ and, at each time step, we do not mask out any position but we generate for all positions. This strategy is designed to save on computation. However, we found that it tends to get stuck in non-fluent sentences and it cannot recover from them.

Examples from all sampling techniques are reported in Table 1.

2.2. BERT's architecture

BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in [13]. This model has an encoder-decoder structure, which is shown in figure 1a.

Transformers are model architectures that eschew recurrence typical of RNNs, relying entirely on attention mechanisms to capture global dependencies between input and output.

2.2.1 Encoder

The encoder consists of a stack of N identical layers. Each layer has two sub-layers. The first is a **multi-head self-attention** mechanism (presented in 2.2.2) and the second is a simple **position-wise fully connected feed-forward** network, which consists of two linear transformations with a *Gaussian Error Linear Unit (GELU)* ([8]) activation in between instead of the original ReLU. We employ a residual connection around each of the two sub-layers, followed by layer normalization.

As shown in Figure 1a, before entering the encoder, input tokens are mapped in a vector of size d_{model} by a **learned embedding**. This component associates to each token a continuous vector called *word embedding*, that captures various attributes of the word, so that words with similar meanings are generally close to one another in the embedding space.

Then, this vector is added to the output vector of the **positional encoder**. Since this model does not contain any recurrence or convolution, positional encoders are used to inject some information about the position of the tokens in the sequence. Since we took d_{model} even, the positional encoder used here is the *sinusoidal positional encoder*.

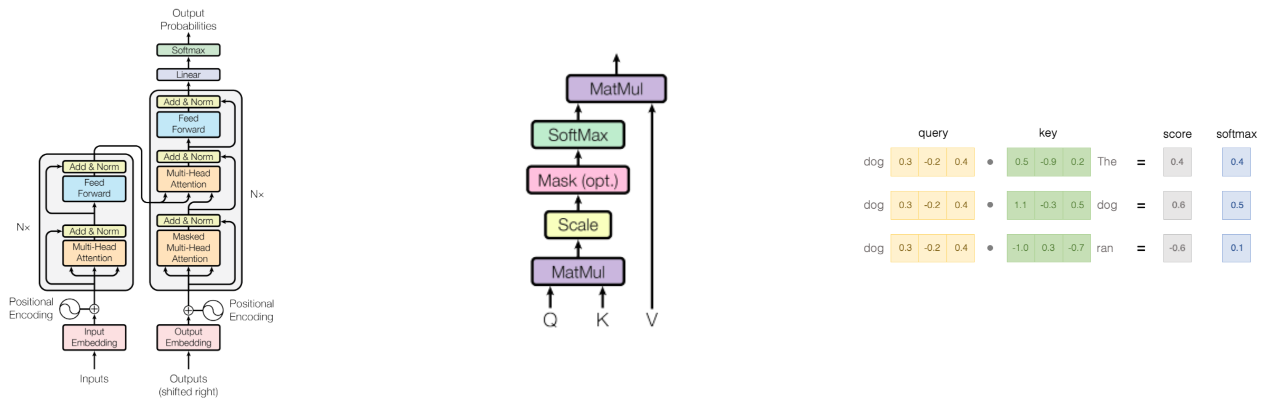
2.2.2 Multi-head attention

In the previous step, we converted each token to a continuous vector which captures various information about this word, but to fully comprehend a language it is not sufficient to understand the individual words that make up a sentence. The attention mechanism enables the model to understand how the words relate to each other in a sentence, producing composite embeddings (weighted averages). To compute these weights, BERT uses the **Scaled Dot-Product Attention** shown in Figure 1b, which assigns to each pair of words a score indicating how strongly they should attend to one another. To prevent the possibility that the gradient of the *softmax* attains too small values, dot products are scaled a by $1/\sqrt{d_k}$ factor.

BERT actually learns multiple attention mechanisms, called *heads*, which operate in parallel. Multi-head attention enables the model to capture a broader range of rela-

Table 1: Sentences generated by BERT's sampling techniques, with temperature =1.0.

Parallel Sequential:	
	head coach : lee dong - won assistant coach : kwon taeyeon by korea institute of football coaches scheme . awarded red card 2016 for former infirm and mentally ill players by korea football association .
	they walked toward st . cloud , hoping that " la petite gourde " would arrive and ask more questions about the battlefield . the traverse - marie journal carried an account of all the sightings .
Sequential:	
	and an old age heart of a different sort , the same feeling , and his own , and of a different kind , a new heart , of a certain sort , perhaps almost both his own kind .
	some more ... some more ... more ... more ... more ... more ... more ... more ... more ... more ... more ... again ... more ... ... all ... again ... ... that was it again .
Parallel:	
	. . . . . . let a . call both a - and an . - that - is and , and , , - in - a , and and and . . . .
	— — a - in — — a . ne / in - to : " and and ... and " : — for all - and and them - to — for all all and - to . —



(a) Transformer architecture.

(b) Scaled Dot-Product attention mechanism.

(c) Example of how to compute weights.

Figure 1: The figure 1c is taken from here, while the others are taken from [13].

tionships between words. Because the attention heads do not share parameters, each head learns a unique attention pattern.

2.2.3 Decoder

The decoder is also composed of a stack of N identical layers. In addition to the two sub-layers of the encoder previously described, the decoder has a third sub-layer, the so-called **masked multi-head attention**. It is placed at the beginning of the stack. This masking, together with the fact that the output embeddings are offset by one position, ensures that the predictions for position i depend only on the known outputs at positions less than i.

Finally, we decided to use *BERT base* configuration instead of the more powerful *BERT large* in order to perform a more meaningful comparison, in that it is closer in size to OpenAI GPT than BERT large.

3. Additional models

In the following, we introduce the models that we compared BERT with.

3.1. OpenAI Generative Pre-Training Transformer (GPT)

OpenAI GPT is a state-of-the-art model on a suite of diverse language tasks. GPT is trained on BooksCorpus (800M words). Its architecture is based on a modification of the Transformer in ([13]) that drops the encoder module and combines the input and output sequences into a single "sentence". It is trained as a standard language model. Model specifications largely follows the original transformer work. For the activation function, GELU is used instead of ReLU, like in BERT-base, while a learned positional embedding is employed instead of the sinusoidal version proposed in ([13]).

Regarding tokenization, GPT performs a pre-tokenization phase where it resorts to the *ftfy* library to clean the raw text in BooksCorpus and to standardize some punctuation and whitespaces and to the *spaCy* tokenizer. The actual tokenization process is done by using the subword tokenizer Byte-Pair Encoding (BPE) ([7]).

3.1.1 Text generation with OpenAI GPT

We used a left-to-right sampling method to generate text from this model. Sentences are built token by token by sampling at each step at random one of the current top k probable tokens. Finally, the generation process is stopped when the maximal length T is reached. It may happen that GPT generates an incomplete sentence.

3.2. Transformer-XL (TFXL)

Transformer-XL ([4]) is a neural architecture that allows to learn token dependencies without disrupting temporal coherence. Its structure is almost identical to the one of GPT ([11]), except for the fact that it introduces a recurrence mechanism for two consecutive segments (a number of consecutive tokens), similarly to what a basic RNN with two consecutive inputs does. Tokenization is performed by resorting to a simple word tokenizer (space and punctuation tokenization). Then, tokens are sequentially fed to the model and at each step the current input is concatenated with the hidden state of the previous segment in order to compute the attention scores. This mechanism enables the model to retain past information. This new implementation requires a relative positional encoder to be used, so that the model can better capture the positional difference that elapse between two distinct words in a segment. The model was trained on the Wikitext-103 corpus.

3.2.1 Text generation with Transformer-XL

We use a left-to-right sampling method to generate text also in this case. The generation process is very similar to the previous one and it is stopped whenever the '<eos>' token gets sampled or the maximum length is reached. If the maximal length is reached and no '<eos>' is met, the model generates an incomplete sentence, as in the case of GPT.

Notice that if the maximal length is reached, TFXL generates T words, while BERT and GPT generate T tokens: this is due to the different tokenizers adopted.

Although only trained on 100M tokens, Transformer-XL is strong at generating long text articles. Notice that:

- it is able to structurally maintain the sectional arrangement of Wikipedia;
- Transformer-XL manages to semantically stay on the same topic throughout the course of generation;
- long-range references are common in the generated text;
- Transformer-XL often generates novel content that is not present in the training data.

Finally, we want to highlight that, before generating a sentence, the model is fed with a starting input. This is done because otherwise Transformer-XL would return very scarse results.

Examples of sentences generated by the aforementioned models are shown in Table 2.

4. Experiments

We resorted to some of the Texygen metrics ([17]) to evaluate generation quality (**BLEU** and **Embedding similarity**) and diversity (**self-BLUE**, **unique *n*-grams**). To make a full comparison with the results in [14], we also computed **perplexity**. Notice that we did not evaluate our models with the NLL metrics implemented in Texygen, because BERT does not provide the estimates of $p(w_i|\text{history})$ on which the NLL metrics are based. Finally, all the metrics were computed by considering the whole corpora all at once and not sentence-wise and we used WikiText-103 as reference text.

4.1. Quality

- **BLEU** It counts the number of matches between the n-grams of the generated text and the n-grams of some reference text. It can be computed as:

$$\text{BLUE} = \text{BP}\exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1}$$

where p_n, the *modified n-grams precision*, is a term rewarding generated texts using n-grams that appear also in the reference text but penalizing the ones resorting to a certain n-gram too many times, N is the number indicating the maximum length of n-grams taken into consideration ($N = 4$ in our case), w_n is the weight associated to n-grams (in our case they are uniform) and BP is the *brevity penalty* ([9]).

We resorted to one of the smoothing technique (**method1**) reported here as in Texygen. If m_n is the number of matched n-gram for a portion of text, this function simply assigns a small value ϵ ($\epsilon = 0.1$ in our case) to m_n when this should be equal to 0, for $n = 1, \ldots, N$.

Notice that we computed the so-called *Corpus-BLEU* metric. As pointed out here, instead of averaging the sentence level BLEU scores (i.e. macro-average precision), the Corpus-BLEU metric accounts for the micro-average precision (i.e. summing the numerators and denominators for each hypothesis-reference(s) pairs before the division).

Table 2: Sentences generated by the other models. In TFXL, the writing in italics represents the seed used to generate the next sentence.

OpenAI GPT:	
	she 'd learned that he used words for the pleasure she was giving him . he let her touch him and wondered what her body looked like . i wish i could see it ... but he
	i want to be alone . i would rather avoid the people who know me only by breathing in their fear . do n't they want to know what it 's like to lose friends that they 've known for
TFXL:	
	Tikal was not sacked but its power and influence were broken. " The biggest problem in the history of the world is the absence of any meaningful representative representative parliamentary representation , " said Professor Mark R. Harris . " There is only one representative representative in the Parliament ,
	The island and its surrounding seas harbour diverse populations of wildlife. The number of animal species on the island is estimated to be between 4 @,@ 900 and 7 @,@ 000 , most of whom are threatened by human intervention .
MLE:	
	oxford felt that it is strengthened , along by the supplier of the hand west of his daughter (debut , and dancers recorded " , was , with land to be dangerous .. ;
	chinese welding also in a series is featured in the traditional tree @-@ mouthed mamba , is something .

- **Perplexity** Given a language model p_M, its perplexity on a certain token sequence $X = (X_1, \dots, X_T)$ is defined as

$$\mathrm{PPL}_{p_M}(X) = \left(\frac{1}{p_M(X_1, \dots, X_T)}\right)^{\frac{1}{T}}, \quad (2)$$

PPL quantifies the ability of a model to predict test data and since we're taking the inverse probability, a lower perplexity indicates a better model.

We measure the perplexity of the samples generated by our models by resorting to the additional language model *transformer_lm.wiki103.adaptive*[1] ([1]) pre-trained on the Wikitext-103 corpus, instead of the *Gated CNN* used in [14].

We report two distinct type of perplexity:

1. Following the approach shown here, standard PPL calculated on a corpus C is the average of all the perplexities retrieved by applying equation 2 to all the m sentences in C.
2. *Corpus-PPL*, instead, is equal to:

$$\mathrm{PPL}_{p_M}(\mathrm{C}) = \sqrt[N]{\frac{1}{\prod_{i=1}^{m} \mathrm{PPL}_{p_M}(X^i)}} \quad (3)$$

where X^i is the i-th sentence in C and N is the total number of words in the entire corpus. Corpus-PPL implies assuming sentences are mutually independent.

- **EmbSim** ([17]) aims to quantify the similarity of two distinct written texts by comparing their word embeddings. For each word embedding, we compute its cosine distance with respect to the other words and then formulate it as a matrix W, where $W_{ij} = \cos(e_i, e_j)$, with e_i, e_j, being the world embeddings of words i and j from real data. W is called **similarity matrix** of the real data. The similarity matrix W' of the generated data is defined analogously. The EmbSim is then defined as

$$\mathrm{EmbSim} = \log\left(\sum_{i=1}^{N} \frac{\cos(W'_i, W_i)}{N}\right), \quad (4)$$

where N is the total number of words and W_i and W'_i denote the i-th column of W and W', respectively.

[1]Code and pre-trained model at https://github.com/pytorch/fairseq.

Notice that we computed the embedding similarity and not the dissimilarity, as in Texygen code.

4.2. Diversity

- **Self-BLEU** It aims to measure the *diversity* of the generated text. Given a generated sentence, it is computed as the average value of its BLEU scores against all the other generated sentences. Sentences using a wide variety of words are associated to low self-BLEU values.
- **Unique n-grams percentage** It is the percentage of distinct generated n-grams that do not appear in the reference text.

4.3. Methodology

In order to replicate the experiment in [14], we generated 1000 sentences for BERT-base, OpenAI GPT and Trasformer XL of maximum length 40. We also chose top k = 100 for both BERT and GPT, while we chose top k = 40 for TFXL, like in [4]. We used the [CLS] starter token as seed for BERT and the empty string for GPT. About Transformer XL, we chose to use a not-empty seed text to reach better performance. To this end, we sampled 1000 sentences from the validation split of Wikitext-103 corpus. Like in [14], we used the parallel-sequential sampling scheme for BERT's generation process. As a baseline, we used 1000 sentences extracted from the Wikitext-103 training set and computed all the metrics against it. Finally, we used the same reference text "wiki103.5k.txt" used in [14].

Note that, we tuned the sampling hyper-parameters (e.g. temperature) for BERT and GPT in order to choose the value that allows us to replicate the experiments in [14]. All the generated text are available at our GitHub repository.

5. Results

The results of our experiments are reported in Tables 3, 4. First of all, the Texygen metrics and the n-grams measures are fully consistent with those in [14]. In fact, the difference between two values provided for the same measure is always below 5%. The same does not hold true for perplexity. The two values we computed are pretty far from those in [14] for some models. This is mainly due to the fact that we computed perplexity by resorting to a different model. We want to stress the fact that we could not resort to the model in [5] as in [14], because we did not found any pre-trained version of it and our lack of resources did not allow us to train it by our own.

We can notice that the BERT and GPT's BLEU scores are still quite low. On the contrary, TFXL is the model that returns the best BLEU score by far, but also the lower perplexity. Nevertheless, it is important to underline that the latter model was fed with an initial input at each sentence generation, otherwise it would have returned very poor results.

We find that BERT generations are more diverse than both GPT and TFXL generations. GPT has higher n-gram overlap (smaller percentage of unique n-grams) with WikiText-103 than BERT, TFXL and WT103, despite being trained on different data. This suggests that GPT, which has also the highest value for Self-BLEU and the smallest value for EmbSim, mainly produces generic sentences.

We want to mention that EmbSim is based on word2vec and skip-grams, a generalization on n-grams. This is why EmbSim and unique n-grams values are coherent.

Finally, we also collected three human judgments on sentence fluency for 100 samples from each model using a four point Likert scale (1 = Not fluent, 2 = Somewhat fluent, 3 = Quite fluent, 4 = Fluent). For each sentence, we asked the annotators to rate the sentence on its fluency and we took the average of the three judgments as the sentence's fluency score. An histogram reporting the result is shown in Figure 2. We got the following mean and standard deviation scores:

- BERT base: mean = 1.9774, std = 0.8054;
- OpenAI GPT: mean = 2.5635, std = 0.5495;
- TFXL: mean = 2.1830, std = 0.7997.

Notice that this results differ from the ones in [14]. In our case, the only model that has a unimodal distribution is GPT. In fact, both BERT and TFXL show a bimodal distribution. Moreover, we can notice that BERT's mean is not within a standard deviation from GPT's mean. However, it is worth to mention that all the annotators noticed that GPT's generation was made of quite generic and simple sentences, as the scores previously shown already suggested us.

The differences in the scores might be due to the fact that the annotators are not English native speakers (although each of them has a CERF level at least equal to C1).

6. Conclusion

In this work we replicated the implementation described in ([14]) and we compared the sentences produced by applying this novel sampling technique to BERT with the ones generated by OpenAI GPT ([11]) and Transformer-XL ([4]). Moreover, we also analyzed through Texygen metrics ([17]) computed by taking as reference the WikiText-103 corpus the texts we obtained.

The value of the metrics we reported and the annotators' judgements suggest that BERT's sentences are not very fluent. Furthermore, unlike GPT and TFXL, it can generate sentences whose length is fixed by the user, but we were not able to let it generate meaningful variable length sentences. Clearly, this represent a significant limitation to its applicability. Indeed, while building our Twitter platform (A), we did not manage to take into consideration the limitation in the number of characters imposed by Twitter.

Finally, because of our scarse resources and the great power required to train text generation models, we were very limited in choosing the models to use for comparison purpose. We tried to train the GAN models provided by Texygen ([17]), but completing this operation would have taken us an enormous amount of time. As an example, we spent 15 hours to perform the vanilla MLE training ([17]) over 10000 sentences randomly extracted from the train split of WikiText-103 in local. Nevertheless, we reported the results obtained by training MLE in Tables 3, 4 and some examples of the sentences it generated in Table 2 anyway.

We released our code at GitHub.

7. Acknowledgements

We deeply thank Simone Masoero, Francesca Miele and Antonio Pellico for evaluating the fluency of the generated sentences.

Table 3: Quality measures.

Model	Corpus-BLEU WT103	EmbSim WT103	PPL	Corpus-PPL
BERT base	9.0017	-2.1656	289.1632	212.9936
GPT	11.1340	-1.7255	175.7630	155.2120
TFXL	22.0293	-2.2129	125.6344	66.6718
WT103	16.1486	-2.4209	186.1305	66.3768
MLE	11.5018	-2.3916	3285.5876	1649.8054
WT103MLE	13.3151	-2.3850	129.5462	71.1355

Table 4: Diversity measures.

Model	Self-BLEU	% Unique n-grams					
		Self			WT103		
		$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$	$n = 4$
BERT base	8.7787	62.7488	92.5906	98.4108	58.5522	91.8973	98.6718
GPT	39.3600	31.5774	67.5123	87.6764	33.6686	73.2306	91.2346
TFXL	22.1123	48.1990	82.0207	95.0062	38.6328	78.1598	94.5532
WT103	10.1783	69.1396	93.8131	98.8779	56.3098	88.4135	97.6938
MLE	10.2983	65.4716	94.0357	99.3602	57.5716	91.2157	99.0350
WT103MLE	10.0048	69.3997	94.1645	98.9331	56.9304	88.4208	97.8288

References

[1] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019.

[2] Valerio Basile, Mirko Lai, and Manuela Sanguinetti. Long-term social media data collection at the university of turin. page 1–6, 2018.

[3] Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. DataStories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[5] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[7] Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38, 1994.

[8] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[9] Kishore Papineni, S. Roukos, T. Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[10] Marco Polignano, Pierpaolo Basile, M. Degemmis, G. Semeraro, and Valerio Basile. Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets. In *CLiC-it*, 2019.

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[12] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, Nov. 2020. European Association for Machine Translation.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[14] Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094, 2019.

[15] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[16] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724, 2015.

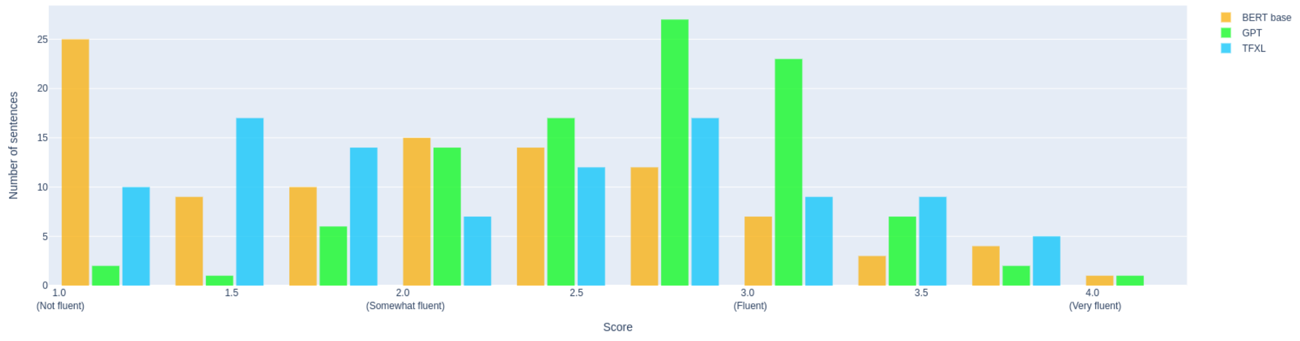


Figure 2: Fluency scores for 100 sentences sampled from BERT base, GPT and TFXL, assigned by human annotators according to a four-point Likert scale.

[17] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *CoRR*, abs/1802.01886, 2018.

A. Twitter platform for Italian users

For our additional task, we decided to exploit the sequential sampling technique presented in section 2.1.1 in order to develop an interactive platform that suggests Italian Twitter users how to complete the tweet they are typing and then translate it to English.

To this end, we used AlBERTo ([10]). This model was pre-trained on 200 million tweets written in Italian randomly extracted from TWITA [2] (no re-tweets).

In order to tailor tweets to BERT's input structure, the authors of [10] carried out a pre-processing phase in which they resorted to the Ekphrasis [3] library. Ekphrasis is a popular tool comprising an NLP pipeline for text extracted from Twitter. It has been used for:

- Normalizing URLs, emails, mentions, percents, money, time, dates, phone numbers, numbers, emoticons. The normalization phase consisted in replacing each term with a fixed tuple $<\ [entitytype]\ >$. In order to facilitate the detokenization phase, we replaced this tuple with $<\ \#\#[entitytype]\ \ \#\#>$. The same holds for the hashtags.

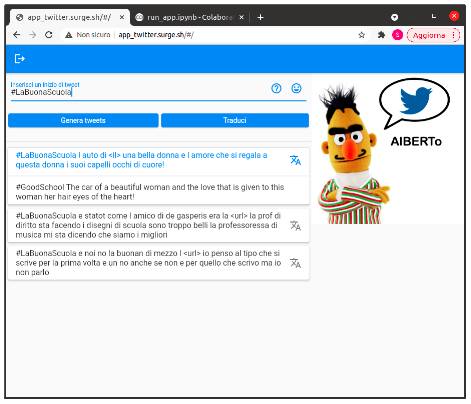
- Tagging and unpacking hashtags. The tagging phase consists in enclosing hashtags with the tags $<\ hashtag\ > \ldots <\ /hashtag\ >$, representing their head and tail in the sentence. Whenever possible, hashtags were unpacked into known words.

In addition to this, we included a post-processing phase to take hashtags back to their original form *#Parola1Parola2...*. Normalization terms, instead, are left as $<\ [entitytype]\ >$ in order to allow users to personalize their own tweets.

For each generated sentence, the platform provides its English translation. This is performed by resorting to the Italian-to-English translator of the OPUS-MT project ([12]). This model is based on a state-of-the-art transformer-based neural machine translator (NMT), namely Marian-NMT 2. Its architecture is very close to the one of BERT'S transformer described in [13]. In order to translate a tweet, we provide as input to this model the pieces of the normalized tweet sentence with unpacked hashtags obtained by separating the entire sentence by emoticons, which are not included in the translator vocabulary.

The platform was implemented with Flutter and it is connected to Google Colab in order to exploit Google's GPUs to accelerate the generation process.

A video teaching you how to use the platform is available at our GitHub repository.



#LaBuonaScuola

#LaBuonaScuola l auto di <il> una bella donna e l amore che si regala a questa donna i suoi capelli occhi di cuore!

#GoodSchool The car of a beautiful woman and the love that is given to this woman her hair eyes of the heart!

#LaBuonaScuola e statot come l amico di de gasperis era la <url> la prof di diritto sta facendo i disegni di scuola sono troppo belli la professoressa di musica mi sta dicendo che siamo i migliori

#LaBuonaScuola e noi no la buonan di mezzo l <url> io penso al tipo che si scrive per la prima volta e un no anche se non e per quello che scrivo ma io non parlo

Figure 3: Twitter app for Italian users.