information

Article

Contextualizer: Connecting the Dots of Context with Second-Order Attention

Diego Maupomé and Marie-Jean Meurs *

Department of Computer Science, Faculty of Sciences, Université du Québec à Montréal, Montreal, QC H3C 3P8, Canada; maupome.diego@courrier.uqam.ca
* Correspondence: meurs.marie-jean@uqam.ca

Abstract: Composing the representation of a sentence from the tokens that it comprises is difficult, because such a representation needs to account for how the words present relate to each other. The Transformer architecture does this by iteratively changing token representations with respect to one another. This has the drawback of requiring computation that grows quadratically with respect to the number of tokens. Furthermore, the scalar attention mechanism used by Transformers requires multiple sets of parameters to operate over different features. The present paper proposes a lighter algorithm for sentence representation with complexity linear in sequence length. This algorithm begins with a presumably erroneous value of a context vector and adjusts this value with respect to the tokens at hand. In order to achieve this, representations of words are built combining their symbolic embedding with a positional encoding into single vectors. The algorithm then iteratively weighs and aggregates these vectors using a second-order attention mechanism, which allows different feature pairs to interact with each other separately. Our models report strong results in several well-known text classification tasks.

Keywords: natural language processing; neural networks; attention mechanism; representation learning

Citation: Maupomé, D.; Meurs, M.-J. Contextualizer: Connecting the Dots of Context with Second-Order Attention. *Information* **2022**, *13*, 290. https://doi.org/10.3390/info13060290

Academic Editor: Diego Reforgiato Recupero

Received: 15 May 2022
Accepted: 3 June 2022
Published: 8 June 2022

1. Introduction

The representation of natural language utterances is a central issue of the application of machine learning techniques to natural language. Indeed, natural language occurrences are difficult to represent using the mathematical objects on which algorithms may operate. Manually constructed symbolic representations tend to "leak", to be unable to capture edge cases, leading to the favoring of learned representations [1,2]. However, it is difficult to learn compact, efficient representations. Language is very sparse: not only does a vast array of local patterns exist, but they will often combine in very few and variable ways. This is not an issue of the granularity of fragmentation. For example, one could choose to break up sentences into characters rather than words. In doing so, the vocabulary of base tokens is greatly reduced in number but data grow sparser. More importantly, larger patterns remain difficult to sanction. That is, whether one opts to break sentences up into characters or words, words will still exist, and only a select few combinations thereof will be conceivable, fewer still will be observed.

As such, it is difficult to establish ways in which to construct suitable utterance representations from base components. Some, such as bag-of-words representations will opt to be deliberately simplified and eliminate the need for the learning of this construction. Other, more complex ones, such as topic models, will allow for some learning of utterance representations but require dedicated learning objectives. In contrast, through backpropagation, neural network approaches allow the learning of complex procedures for constructing utterance representations while still allowing for variety in downstream training objectives. Regardless, while the exact parametrization of these operations is to be inferred from the data, there is still considerable structure to provide. That is, the operations

in the neural network and their composition need to be specified. Some aspects of this structure are somewhat imposed by practical issues, such as the ability to handle variable length data. Other, more deliberate choices are informed by prior beliefs about language. For example, tree-structured recursive neural networks are predicated on the notion that complex sentences are recursively constructed from their parts [3,4].

As mentioned, language is sparse and combinatorially difficult: even in highly synthetic languages, combinations of words will carry some semantic sense that the words themselves do not carry. This can be broadly construed as an issue of *context*: parts of utterances need to be put into the context of the whole in order to be understood.

Contextualization is fundamentally difficult because it is a circular problem. One cannot recompose a whole by putting its parts in the context of said whole without already knowing what the whole is. One potential solution to this is to iteratively adjust the context against which the parts are compared. That is, to begin with a presumably erroneous value of the context representation and adjust this value with respect to the tokens at hand. This is what Transformer encoders [5] do: First, the tokens in a sequence are compared against each other by the self-attention mechanism [6]. This consists of having each token *attend* over all tokens in the sentence, with a parametric attention function producing a scalar weighting of the importance of each token with respect to the attender. Then, a new representation for each token is produced through this weighting. The process is then repeated a set number of times, as shown in Figure 1. In doing so, the word representations produced by this encoder are put into the context of the whole. Transformers have achieved much success in various Natural Language Processing (NLP) tasks [7–10], ranging from sentiment analysis to question-answering and natural language inference [11].

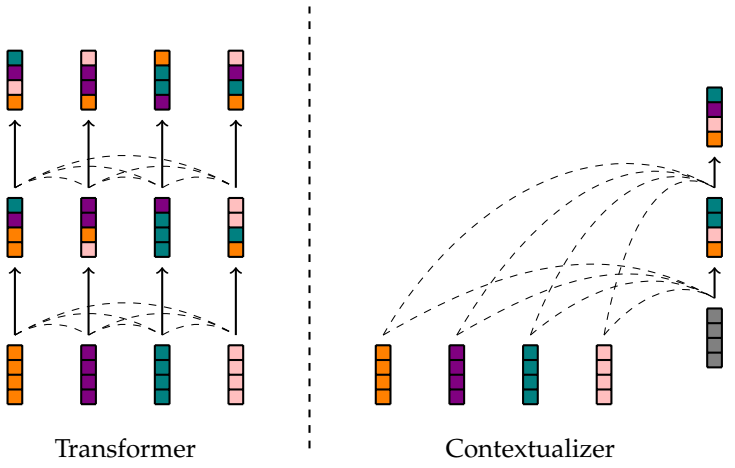

Figure 1. The Transformer encoder updates the representations of tokens with respect to each other over a set number of steps by letting each token attend over all tokens in the sequence. This amounts to a distributed representation of the context. The proposed approach updates a single context vector which attends over the tokens at hand.

Nonetheless, the self-attention mechanism on which Transformers are built has two chief disadvantages. Firstly, because all word pairs are evaluated, the complexity is quadratic with respect to the length of the utterance. Secondly, the weighting provided by the self-attention mechanism is based on bilinear forms, mapping each pair of word vectors to a single scalar. As such, Transformers require multiple sets of self-attention parameters, called *heads*, so that separate heads might focus on different features of the word vectors. To address these issues, we propose a new architecture—the *Contextualizer*—based on iteratively adjusting a context vector using a second-order attention mechanism. Its computational complexity grows linearly with respect to the sequence length, as opposed to quadratically.

This article is organized as follows. Section 2 introduces the proposed approach. Section 4 describes experiments conducted in a few well-known document classification tasks and the results obtained. Finally, Section 5 concludes this article.

2. Contextualizer

The proposed encoder utilizes the approach illustrated in Figure 1. It proceeds as follows: Over a set number of steps, token representations are matched to the context representation to produce a contextualized representation. These representations are then aggregated into a new context representation, and the process begins anew. The present section details how these computations are carried out in a general-purpose setting.

Let $w_1, \ldots, w_n$ be a sequence of tokens forming a document of length n, indexed by $i = 1, \ldots, n$. Before the contextualization steps, each token is mapped to a single real vector combining information about its identity and position in the sequence. The former is provided by a symbol embedding (e.g., pretrained word vectors) of dimension m, $\boldsymbol{e}(w_i) \in \mathbb{R}^m$. The latter is based on a positional encoding inspired by Maupomé et al. [12]. This positional encoding is as follows: given a vector of parameters, $\boldsymbol{s} \in \mathbb{R}^m$, the jth component of the encoding for position i, $\boldsymbol{p}(i)$, is given by:

$$\boldsymbol{p}(i)_j = \frac{\exp(is_j)}{\sum_{i'=1}^{n} \exp(i's_j)}.$$

Multiplicative constants amplify or dampen the peak of a softmax application. By applying the softmax across tokens in the sequence, the parameter vector $\boldsymbol{s}$ allows the model to modulate certain positions for different components of $\boldsymbol{p}$. Combining these two aspects, the token and its position, the vector representation of token w_i is:

$$\boldsymbol{x}_i = \boldsymbol{e}(w_i) * \boldsymbol{p}(i).$$

where $*$ denotes the Hadamard product.

Next, there are K contextualization steps, indexed by $k = 1, \ldots, K$. Each of these steps will produce a new context vector, $\boldsymbol{c}^k$. The default context used at the first step of contextualization, $\boldsymbol{c}^{(0)}$, can be set to a constant or a learned parameter, for example. This context vector will contextualize the tokens, which will then be aggregated into a new context. An attention mechanism provides the contextualizing function called at every iteration. Using any of the various attention mechanisms in the literature, contextualizing each token would amount to producing a scalar weight, α_i, for each token depending on its content and that of the attender (the context vector in our case). The contextualization of token $\boldsymbol{x}_i$ at step k with respect to the previous context, $\boldsymbol{c}^{(k-1)}$, would then be

$$(\boldsymbol{c}^{(k-1)}, \boldsymbol{x}_i) \mapsto \alpha_i^{(k)} \boldsymbol{x}_i.$$

However, the use of scalar attention weights requires that each component in the operands interacts only with its homolog, collapsing all information to a single number. One must therefore compute several of these interactions with different sets of parameters—called attention heads—so that each of these may focus on different features. This is particularly important when using distributed token representations, where each component might carry a different semantic sense. As such, Transformers contain several attention heads. In contrast, to have the weight of each token be a vector, $\boldsymbol{\alpha}_i$, rather than a scalar, α_i, would let each component of the token representation have a separate salience with respect to the current context:

$$(\boldsymbol{c}^{(k-1)}, \boldsymbol{x}_i) \mapsto \boldsymbol{\alpha}_i^{(k)} * \boldsymbol{x}_i, \tag{1}$$

This is illustrated in Figure 2.

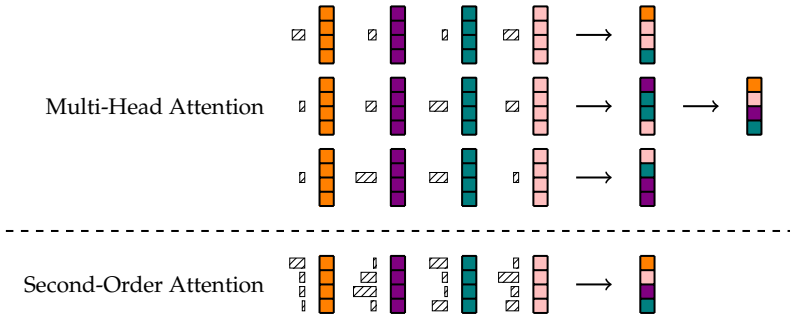

Figure 2. Scalar attention requires the use of different sets of attention parameters—heads—to attend to different features of the tokens being aggregated; vector attention allows features to be weighted independently.

Such a mechanism would eliminate the need for several heads, as each feature of each token can interact with each feature of the context by a different parameter. However, a second-order attention weighting would require parametrization by a tensor of degree three (3), which would take the parameter count of the model to O (m^3), as both the input and the context vectors are of dimension m. For token representations of even modest size, this would result in a computationally intensive model. Moreover, if the transformation to be learned does not require a full-rank degree-three tensor, such a parametrization would ostensibly be prone to overfitting because of its excess capacity. Instead, a tensor of rank (not to be confused with the degree or order of a tensor, the rank of a tensor is analogous to the rank of a matrix) u can be used, with u becoming a hyperparameter, see [13–15]. Using this approach, the attention vector for token $\boldsymbol{x}_i$ would be computed as:

$$\boldsymbol{\alpha}_i^{(k)} = \mathbf{W}^{(k)}(\mathbf{U}^{(k)}\boldsymbol{x}_i * \mathbf{V}^{(k)}\boldsymbol{c}^{(k-1)}) + \boldsymbol{b}^{(k)},$$

where $\mathbf{U}^{(k)}, \mathbf{V}^{(k)} \in \mathbb{R}^{u \times m}$ and $\mathbf{W}^{(k)} \in \mathbb{R}^{m \times u}$ are the matrices of parameters for the kth contextualization step, and $\boldsymbol{b}^{(k)}$ is the corresponding bias vector.

The newly computed attention vectors serve to update the context vector. This update $\boldsymbol{z}^{(k)}$, is then obtained by adding the contextualized token vectors together, followed by layer normalization [16]:

$$\tilde{\boldsymbol{z}}^{(k)} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i^{(k)} * \boldsymbol{x}_i$$

$$\boldsymbol{z}^{(k)} = \text{LayerNorm}(\tilde{\boldsymbol{z}}^{(k)})$$

The update is then applied to the context vector to obtain the new value for the context vector:

$$\boldsymbol{c}^{(k)} = \boldsymbol{c}^{(k-1)} + \boldsymbol{z}^{(k)}$$

As mentioned, this process is repeated over a set number of steps, allowing information from different sets of tokens to inform the context. The final context vector, $\boldsymbol{c}^{(K)}$, then contains 3a fixed-size summary of the sequence of tokens at hand.

By reducing sequences of arbitrary length to fixed-size encodings, the proposed approach could potentially squash some information, whereas Transformers encode their input into a sequence of vectors. In return, the number of comparisons in one iteration of the Contextualizer algorithm grows linearly with respect to the number of tokens, as opposed to quadratically for the Transformer. Table 1 presents the computational complexities of these two algorithms as well as recurrent and convolutional layers. In addition, as illustrated by Figure 1, the Transformer has the drawback of losing sight of the original representation of the tokens, whereas the Contextualizer does not.

Table 1. Complexities of common layers used in NLP; l designates the kernel size, h, the number of attention heads, n, m and u, designate the length of the sequence, the dimension of the word representations and the multiplicative dimension, respectively. For Transformers and Contextualizers, the complexity is for a single contextualization step.

Layer	Complexity
Recurrent	$\mathcal{O}$ (nm^2)
Convolutional	$\mathcal{O}$ (lnm^2)
Transformer encoder	$\mathcal{O}$ (hn^2m)
Contextualizer	$\mathcal{O}$ (num)

3. Related Work

The computational complexity of Transformers makes them unwieldy for long sequences. As such, there have been several efforts to simplify the computation of full token-to-token self-attention to a lighter, more computationally efficient version.

For example, self-attention can be limited to local neighborhoods [17]. That is, instead of comparing tokens attend to each other throughout the sequence, tokens can be restrained to attending over a local portion of the sequence. This approach can be complemented by having sliding-window neighborhoods [18,19]. It can also be combined with *causal masking*: allowing tokens to attend only over preceding tokens. In doing so, segments can be chained recursively, allowing deeper contextualization levels to receive information from earlier segments [20].

More sophisticated, dynamic approaches can rely on inferred neighborhoods. Rather than determining neighbors by position, tokens can be bucketed by locality-sensitive hashing [21] or clustering [22]. Alternatively, neighborhoods can be determined by the syntax tree of the utterance at hand [23].

Yet another approach is to replace softmax self-attention with a lighter variant. For example, by replacing the exponential kernel implicit to softmax self-attention by a polynomial kernel, key–value products can be shared across queries, reducing computational complexity [24]. In the same vein, softmax self-attention can be approximated via random feature maps, thus reducing the dimension of the attention space as a function of sequence length rather than eliminating token pairs [25].

4. Experiments and Results

4.1. Exploratory Experiments

We began with experiments on the well-known Rotten Tomatoes dataset (MR) [26] (available at https://www.cs.cornell.edu/home/llee/papers/pang-lee-stars.home.html, accessed on 15 May 2022). It consists of 11 k English-language sentences from film reviews classed as either positive or negative in equal proportions. The documents are fairly short, with 95% of them being 45 words long or shorter.

The first experiments sought to compare the test set classification accuracy and computation time of Transformer and Contextualizer models of comparable sizes. These parameter counts were chosen to be relatively small in accordance with the limited size of the dataset. For each architecture, four models of 0.5, 1, 1.5 and 2 million parameters were trained and tested. These counts excluded the initial token embedding layer. Following the Transformer approaches [7,10,27], the documents were tokenized into word-piece tokens [28,29]. The hyperparameters of the models were set following Vaswani et al. [5] while adjusting for the smaller model size. The dimension of the embedding space, m, was set to 128. The number of contextualization steps (the number of encoding layers in the Transformer) was set to five for all models. The number of attention heads for Transformers was set to four. Hence, the variable adjusted to increase the parameter count of the models was the dimension of the attention space for Transformers and the rank of the tensor decomposition, u, for Contextualizers.

Models were trained on a Intel Core i7-7700 machine with 32 GiB of memory over 10 epochs with batches of 32 examples using the Adam [30] optimizer, with a learning rate of 1×10^{-4}. The best model on a 10% validation set over the 10 epochs was selected for testing.

Results are presented in Table 2. As shown, both architectures show comparable results across model sizes. As expected, computation time is much greater for Transformer networks.

We then proceeded with experiments measuring the effect on performance of the nature of the default context. All Contextualizer models shared the same configurations except the default context, which was set to be either a constant, $\boldsymbol{c}^{(0)} = \mathbf{1}$, a vector of learned parameters, $\boldsymbol{c}^{(0)} = \boldsymbol{c}_d$ or a random vector redrawn for every document from a uniform distribution, $\boldsymbol{c}^{(0)} \sim \mathcal{U}(-\mathbf{1}, \mathbf{1})$. We hypothesized that using a random default context would make the network more robust by reducing dependence on prior beliefs and therefore mitigating overfitting. For the same reasons, one could expect a learned default context to be more likely to overfit than a constant one.

Table 3 summarizes the results. As one might expect, a random starting context vector hurts performance when contextualization is performed but once. The models are quick to adjust, as all choices of default context seem to arrive at very similar final accuracies.

Table 2. Test accuracy (%) on the MR task and computation time (ms/batch) for Transformer and Contextualizer models of similar sizes. The parameter counts exclude word-piece embeddings. The batch size is 32.

Parameter Count	Contextualizer		Transformer	
	Accuracy	Time	Accuracy	Time
0.5 M	73.5	57	72.4	118
1.0 M	74.3	96	74.9	164
1.5 M	73.4	120	73.0	223
2.0 M	74.0	151	74.7	265

Table 3. Test accuracy (%) on the MR task for different default context strategies

$\boldsymbol{c}^{(0)}$	K	
	1	**5**
$\mathbf{1}$	73.1	71.2
$\boldsymbol{c}_d$	73.5	72.2
$\mathcal{U}(-\mathbf{1}, \mathbf{1})$	57.9	72.4

4.2. Further Results

We continue with experiments in binary document classification on other well-known English-language datasets in order to compare the performance of the proposed approach to the Transformer-based Universal Sentence Encoder architectures (USE) [8]. The Subjectivity dataset (available at https://www.cs.cornell.edu/people/pabo/movie-review-data/, accessed on 15 May 2022) (SUBJ) [31] comprises 10 k sentences around films classed as subjective or objective, released in June 2004. Annotation is automatic based on whether the sentence is a synopsis (objective) or an appreciation (subjective). The Customer Reviews dataset (CR) (available at http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar accessed on 15 May 2022), introduced by Hu and Liu [32], comprises 3775 reviews of electronic products. These reviews were extracted from Amazon and CNET and manually annotated. They are equally divided between positive and negative sentiment. Finally, the Multi-Perspective Question Answering dataset (available at https://mpqa.cs.pitt.edu/corpora/mpqa_corpus/ accessed on 15 May 2022) (MPQA) [33] deals again in sentiment polarity, containing 10,606 k phrases from press articles.

Test accuracy on several benchmark text classification tasks of our Contextualizer models compared to the Universal Sentence Encoders (USE) [8], Transformer-based (T) and Deep-Averaging-Network-based (D) is shown in Table 4. The results of these experiments demonstrate that the Contextualizer architecture can perform competitively, even with small models and relatively small datasets.

Table 4. Test accuracy (%) on several benchmark text classification tasks of our Contextualizer models compared to the Universal Sentence Encoder architectures (USE), Transformer-based (T) and Deep-Averaging-Network-based (D).

Model	MR	CR	SUBJ	MPQA
Contextualizer	76.6	79.0	91.2	85.3
USE (T)	81.4	87.4	93.9	87.0
USE (D)	74.5	81.0	92.7	85.4

5. Conclusions

We proposed an algorithm for constructing sentence representations based on the notion of iteratively adjusting a central context vector. This algorithm was closely related to the encoder part of the Transformer algorithm. One key difference was the use of the proposed second-order attention mechanism, replacing multiple attention heads.

Another important difference was the computational complexity, which was linear in sequence length. Transformer models have been the driving force behind the expansive use and development of large models such as BERT [7], RoBERTa [27], GPT-3 [10], Electra [9], among others, which are extensively trained by adapted language-modeling tasks. The reduced complexity of the Contextualizer model would be of use both in terms of pretraining and in terms of wielding these large models in downstream tasks. This is important given how the computational cost of these large models can further the economical divide between low- and high-resource laboratories and companies [34]. Additionally, this computational demand also incurs a significant environmental impact [35]. Therefore, lighter-computation approaches could help mitigate these concerns.

Yet, as seen in Section 4, the Contextualizer achieved results comparable to those of a Transformer when controlling for model size. Our approach was also able to achieve competitive results in benchmark document classification tasks even with low parameter counts. Furthermore, our results suggested the approach was robust to different choices of the number of contextualization steps and default contexts. Further work will be conducted in this direction, as well as in formally characterizing the conditions that stabilize the context vector as the number of contextualization steps increases.

Author Contributions: Conceptualization, D.M.; methodology, D.M. and M.-J.M.; software, D.M.; validation, formal analysis, investigation, D.M. and M.-J.M.; resources, M.-J.M.; writing—original draft preparation, D.M.; writing—review and editing, D.M. and M.-J.M.; supervision, M.-J.M.; funding acquisition, D.M. and M.-J.M. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) (MJ Meurs, NSERC grant number 06487-2017), and the support of the Government of Canada's New Frontiers in Research Fund (NFRF), (MJ Meurs, NFRFE-2018-00484).

Data Availability Statement: The Contextualizer source code is available under the GNU GPL v3 license to ensure reproducibility. It can be found in the following repository: https://gitlab.labikb.ca/ikb-lab/nlp/contextualizer, accessed on 15 May 2022.

Acknowledgments: The authors want to thank Fanny Rancourt for proofreading the manuscript and providing constructive comments and suggestions. This research was enabled in part by support provided by Calcul Québec (https://www.calculquebec.ca, accessed on 15 May 2022) and Compute Canada (https://www.computecanada.ca, accessed on 15 May 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Manning, C.; Schutze, H. *Foundations of Statistical Natural Language Processing*; MIT Press: Cambridge, MA, USA, 1999.
2. Ferrone, L.; Zanzotto, F.M. Symbolic, distributed, and distributional representations for natural language processing in the era of deep learning: A survey. *Front. Robot. AI* **2020**, *70*, 153. [CrossRef] [PubMed]
3. Socher, R.; Manning, C.D.; Ng, A.Y. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks. In Proceedings of the NIPS—2010 Deep Learning and Unsupervised Feature Learning Workshop, Whistler, BC, Canada, 10 December 2010; p. 9.
4. Bowman, S.R.; Potts, C.; Manning, C.D. Recursive Neural Networks Can Learn Logical Semantics. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality, Beijing, China, 31 July 2015; Association for Computational Linguistics: Beijing, China, 2015; pp. 12–21. [CrossRef]
5. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
6. Cheng, J.; Dong, L.; Lapata, M. Long Short-Term Memory-Networks for Machine Reading. *arXiv* **2016**, arXiv:1601.06733.
7. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
8. Cer, D.; Yang, Y.; Kong, S.y.; Hua, N.; Limtiaco, N.; John, R.S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. Universal Sentence Encoder for English. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Brussels, Belgium, 31 October–4 November 2018; pp. 169–174.
9. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv* **2020**, arXiv:2003.10555.
10. Floridi, L.; Chiriatti, M. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.* **2020**, *30*, 681–694. [CrossRef]
11. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *arXiv* **2019**, arXiv:1804.07461.
12. Maupomé, D.; Rancourt, F.; Armstrong, M.D.; Meurs, M.J. Position Encoding Schemes for Linear Aggregation of Word Sequences. In Proceedings of the Canadian Conference on Artificial Intelligence, Vancouver, BC, USA, 25–28 May 2021; PubPub: Vancouver, BC, USA, 2021. [CrossRef]
13. Rabanser, S.; Shchur, O.; Günnemann, S. Introduction to Tensor Decompositions and their applications in Machine Learning. *arXiv* **2017**, arXiv:1711.10781.
14. Sutskever, I.; Martens, J.; Hinton, G.E. Generating Text with Recurrent Neural Networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 1017–1024.
15. Maupomé, D.; Meurs, M.J. Language Modeling with a General Second-Order RNN. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; European Language Resources Association: Marseille, France, 2020; pp. 4749–4753.
16. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
17. Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; Tran, D. Image Transformer. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; Proceedings of Machine Learning Research: Stockholm, Sweden, 2018; Volume 80, pp. 4055–4064.
18. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The Long-Document Transformer. *arXiv* **2020**, arXiv:2004.05150.
19. Chelba, C.; Chen, M.; Bapna, A.; Shazeer, N. Faster Transformer Decoding: N-gram Masked Self-Attention. *arXiv* **2020**, arXiv:2001.04589.
20. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv* **2019**, arXiv:1901.02860.
21. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The Efficient Transformer. *arXiv* **2020**, arXiv:2001.04451.
22. Roy, A.; Saffar, M.; Vaswani, A.; Grangier, D. Efficient Content-Based Sparse Attention with Routing Transformers. *arXiv* **2020**, arXiv:2003.05997.
23. Bai, J.; Wang, Y.; Chen, Y.; Yang, Y.; Bai, J.; Yu, J.; Tong, Y. Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees. *arXiv* **2021**, arXiv:2103.04350.
24. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; Daumé, H., III, Singh, A., Eds.; Proceedings of Machine Learning Research: 2020; Volume 119, pp. 5156–5165. Available online: https://proceedings.mlr.press/v119/katharopoulos20a.html (accessed on 1 April 2022)
25. Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. Rethinking Attention with Performers. *arXiv* **2021**, arXiv:2009.14794.
26. Pang, B.; Lee, L. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, Ann Arbor, MI, USA, 25–30 June 2005; pp. 115–124.
27. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.

28. Schuster, M.; Nakajima, K. Japanese and Korean voice search. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 5149–5152, ISSN 2379-190X. [CrossRef]
29. Merity, S.; Xiong, C.; Bradbury, J.; Socher, R. Pointer Sentinel Mixture Models. *arXiv* **2016**, arXiv:1609.07843.
30. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
31. Pang, B.; Lee, L. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, 21–26 July 2004; p. 271.
32. Hu, M.; Liu, B. Mining and Summarizing Customer Reviews. In Proceedings of the KDD'04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; ACM Press: Seattle, WA, USA, 2004.
33. Wiebe, J.; Wilson, T.; Cardie, C. Annotating Expressions of Opinions and Emotions in Language. *Lang. Resour. Eval.* **2005**, *39*, 165–210. [CrossRef]
34. Luitse, D.; Denkena, W. The great Transformer: Examining the role of large language models in the political economy of AI. *Big Data Soc.* **2021**, *8*, 20539517211047734. [CrossRef]
35. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 3645–3650. [CrossRef]