Thinking Tokens for Language Modeling

Anonymous AITP submission

Abstract

How much is 56 times 37? Language models often make mistakes in these types of difficult calculations. This is usually explained by their inability to perform complex reasoning. Since language models rely on large training sets and great memorization capability, naturally they are not equipped to run complex calculations. However, one can argue that humans also cannot perform this calculation immediately and require a considerable amount of time to construct the solution. In order to enhance the generalization capability of language models, and as a parallel to human behavior, we propose to use special 'thinking tokens' which allow the model to perform much more calculations whenever a complex problem is encountered.

1 Introduction

Language models based on neural networks have gained a great deal of interest in recent years [15, 14]. Their impressive and coherent answers amazed people across many industries. However, it has soon been discovered that these language models have problems with complex tasks [5, 4].

Complex questions such as *'how much is 56 times 37'* which are computationally requiring, are problematic for the language model to process or even answer correctly. One can argue that humans also cannot perform this calculation right away and require a considerable amount of time to provide a solution.

Although these reasoning abilities could be improved by employing a large amount of supervision by providing labeled examples to the model [4], we aim for a much faster unsupervised approach.

Figure 1: Illustration of 'thinking tokens' (marked as <T>) in a sentence which requires a complex calculation and the positive impact of this approach on perplexity of the model (lower is better).

To enhance the generalization capability of language models, we propose to use special 'thinking tokens' which allow the model to perform much more calculations whenever a complex problem is encountered. This could result in an improved generalization capability of language models, which could adapt to more complex tasks and even decide themselves what strategy is most beneficial for an encountered problem.

2 Related work

Research regarding reasoning can be traced back to 1959 [16], and now continues to be a big part of theorem proving [18, 6, 8]. Presently, large language models are being used to learn reasoning from natural language [1, 11].

A similar problem has also been studied in [9], where a language model recomputes only a part of the recurrent hidden layer. Another work with a similar motivation explores the possibility of using a neural network that is capable of learning algorithms [10].

3 Thinking tokens for language models

Our approach is to introduce special 'thinking tokens' ($< T >$) after each word in a sentence whenever a complex problem is encountered. The core idea is that each 'thinking token' would buy more time for the model before an answer is expected, which would be used to run additional computations to better answer a complex problem that was presented. This concept has a great potential in recurrent neural networks [3, 7] due to their architecture, because it enables the RNN to perform multiple in-memory operations in a single step, meaning that extra calculations can be run in the hidden layer multiple times.

As a proof of concept, we have added N 'thinking tokens' ($< T >$) after each observed word in a dataset. Our vision is that this basic concept can be extended to a self-adjusting model, which will be able to decide itself if and how many 'thinking tokens' will be used for a specific problem, where N could also vary throughout the sentence. This would allow us to reduce the computational time, which would not increase N times. The visualization of our core idea, which we aim to validate in this paper, is presented in Figure 1.

4 Results

Experiments execution has successfully produced numerous examples where the usage of 'thinking tokens' leads to an improvement in the model's judgment. Preliminary results show that sentences that require non-trivial reasoning, have the biggest improvement in perplexity when 'thinking tokens' are used compared to the standard model. This is also observable on the sample sentences from Maths dataset in Table 2. A larger scope of examples across all the datasets is in Appendix A.1.

We can observe that introduction of 'thinking tokens' is also successful for sentences that include specific numbers or representative symbols of numerical values.

Dataset	**Sentence**	**Ppl. orig. ↓**	**Ppl. <T> ↓**
maths	What is the remainder when 8922293 is divided by 263 ? 18	16.8	13.1
maths	Convert -464 (base 9) to base 6 . -1434	24.3	19.8

Table 1: Examples of sentences where the introduction of 'thinking tokens' is beneficial to the model. First and second column refer to the name of the dataset and the specific sentence. Two rightmost columns refer to the perplexity without tokens (Ppl. orig.) and with 'thinking tokens' (Ppl. <T>). A downward arrow ↓ indicates that lower is better.

5 Discussion and Future work

Language models often make mistakes in complex problems like calculations or reasoning, since they rely on large training sets and their great memorization capability. We show that giving RNNLM more time to 'think' and not pressuring the model to produce an answer immediately, helps the model resolve various complex tasks more accurately.

Building on the proof of concept, we plan to extend our research and create a model that would be able to decide itself how much extra time is needed in order to produce the best answer possible. If successful, this concept could be implemented as a default behavior for language models that encounter complex and computationally demanding tasks. We also believe that the ability of a model to self-regulate this factor would vastly improve adaptability and generalization capability of language models in general.

References

[1] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[2] Russell Cooper and Andrew John. *Macroeconomics: Theory through applications*. Saylor Foundation, 2012.

[3] J Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, June 1990.

[4] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. Mathematical capabilities of chatgpt, 2023.

[5] Timothy Gowers. It's amusing when ChatGPT makes ridiculous mathematical mistakes. But of course, it's more interesting to find out what it can do well. Here's one example that wasn't bad: I gave it a very rough outline of a proof and asked it to fill in the details. https://twitter.com/wtgowers/status/1611750773607604224, 2023. [Online; accessed 2023-02-27].

[6] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughin, Tat Thang Nguyen, and et al. A formal proof of the kepler conjecture. *Forum of Mathematics, Pi*, 5:e2, 2017.

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

[8] Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Een, Francois Chollet, and Josef Urban. Deepmath - deep sequence models for premise selection. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[9] Yacine Jernite, Edouard Grave, Armand Joulin, and Tomas Mikolov. Variable computation in recurrent neural networks, 2016.

[10] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 190–198, Cambridge, MA, USA, 2015. MIT Press.

[11] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022.

[12] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models, 2017.

[13] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. volume 2, pages 1045–1048, 01 2010.

[14] John Naughton. The ChatGPT bot is causing panic now – but it'll soon be as mundane a tool as Excel. https://www.theguardian.com/commentisfree/2023/jan/07/chatgpt-bot-excel-ai-chatbot-tec, 2023. [Online; accessed 2023-02-28].

[15] Kevon Roose. The Brilliance and Weirdness of ChatGPT. https://www.nytimes.com/2022/12/

05/technology/chatgpt-ai-twitter.html, 2022. [Online; accessed 2023-02-28].

[16] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[17] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *ArXiv*, abs/1904.01557, 2019.

[18] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. Malarea sg1 - machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning*, pages 441–456, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

A Appendix

A.1 Experiments

Dataset	Sentence	Ppl. orig. ↓	Ppl. <T> ↓
ptb	britain has two main index-arbitrage instruments	24.1	21.7
ptb	today pc shipments annually total some N billion world-wide	45.5	42.3
wt-2	The discography of LiSA includes three studio albums , one extended play , ten singles , and five video albums	130.1	127.7
wt-2	Complex N lies to the west of the Bat Palace and Temple III . The complex dates to AD 711	246.8	244.2
etb	increase in deficit raises the interest rate	20.5	18.5
etb	however too much money in circulation can lead to inflation	27.9	25.5

Table 2: Examples of sentences where the introduction of 'thinking tokens' is beneficial to the model. First and second column refer to the name of the dataset and the specific sentence. Two rightmost columns refer to the perplexity without tokens (Ppl. orig.) and with 'thinking tokens' (Ppl. <T>). A downward arrow ↓ indicates that lower is better.

To evaluate the plausibility of our idea, we first propose to extend the standard recurrent neural language model with extra tokens. This does not require any change in the model architecture and can be achieved by modifying the input data by adding N 'thinking tokens' after each word. In our case $N = 1$.

We have chosen a simple setup where a RNN LM with one hidden layer is used. We train a baseline model, standard LSTM LM [7], and finally a model with the 'thinking tokens', as we believe the perplexity differences could be rather small. After all, the mistakes in reasoning about numbers influence entropy much less than for example correctly capturing uni-gram frequencies of the most common words.

We have designed an experiment to identify sentences in which the largest differences in perplexity between the two models can be observed. This could allow us to determine in which cases the usage of 'thinking tokens' is beneficial for the model. For the purpose of fair results evaluation, the loss generated by 'thinking tokens' is omitted from the calculation of perplexity.

Models were trained on standard language modeling tasks like Penn TreeBank [13], WikiText-2 [12] and also on mathematics dataset [17] and dataset retrieved from MacroEconomics textbook [2]. Hyper-parameters and additional experiments are listed in A.5.

A.2 Word probabilities

To give the reader more insight into what happens when the 'thinking token' is used, we have decided to show the probabilities for each word in two sample sentences. It is important to note that the probabilities of 'thinking tokens' are omitted.

'What is the remainder when 8922293 is divided by 263 ? 18'.

Word: What
LSTM: 0.27570505869594907016
LSTM+<T>: 0.27460685731534064
Word: is
LSTM: 0.9983407855033875
LSTM+<T>: 0.9994571208953857

Word: the
LSTM: 0.7941651940345764
LSTM+<T>: 0.7810274958610535
Word: remainder
LSTM: 0.05905058979988098
LSTM+<T>: 0.39061781764030457
Word: when
LSTM: 0.9977582693099976
LSTM+<T>: 0.9991976022720337
Word: 8922293
LSTM: 8.769490705162752e-06
LSTM+<T>: 2.8874606869067065e-05
Word: is
LSTM: 0.9854738712310791
LSTM+<T>: 0.977165162563324
Word: divided
LSTM: 0.9997884631156921
LSTM+<T>: 0.9993095993995667
Word: by
LSTM: 0.9998854994773865
LSTM+<T>: 0.9999291300773621
Word: 263
LSTM: 3.43535648426041e-05
LSTM+<T>: 4.3290932808304206e-05
Word: ?
LSTM: 0.9997261762619019
LSTM+<T>: 0.9896721243858337
Word: 18
LSTM: 0.02015523798763752
LSTM+<T>: 0.019233182072639465

'Increase in deficit raises the interest rate'.

Word: increase
LSTM: 0.00013779969594907016
LSTM+<T>: 0.0005539595731534064
Word: in LSTM: 0.7479172348976135
LSTM+<T>: 0.723701536655426
Word: deficit LSTM: 0.0001904059899970889
LSTM+<T>: 0.00011684149649227038
Word: raises LSTM: 0.01803828589618206
LSTM+<T>: 0.004907318856567144
Word: the LSTM: 0.46662768721580505
LSTM+<T>: 0.4531695246696472
Word: interest LSTM: 0.05833666771650314
LSTM+<T>: 0.07482223212718964
Word: rate LSTM: 0.9725480079650879
LSTM+<T>: 0.9811777472496033

A.3 Perplexity of models

In Table 3 we have evaluated language models on 4 datasets. In the first column, we have standard LSTM with 1 layer while in the second column, we have results for the same LSTM, but with a 'thinking token' after each observed word. It is important to note that loss from 'thinking tokens' was not included in the calculation of perplexity.

It could be observed that addition of 'thinking token' results in slight performance decrease in perplexity. However, the main goal of $< T >$ tokens is not to improve perplexity, but to enhance model's capability to 'think'.

Validation perplexity		
Dataset	**LSTM**	**LSTM+<T>**
Penn tree bank	68.2	68.4
Wikitext-2	76.8	82.4
Economic textbooks	49.0	51.4
Maths	19.8	19.8

Table 3: 4 datasets ptb [13], wt-2 [12], etb [2], maths [17].

A.4 Number of thinking tokens

Validation perplexity		
Dataset	**LSTM+2<T>**	**LSTM+3<T>**
ptb	71.3	78.6
wt-2	89.1	94.0
etb	56.2	60.9
maths	21.0	24.4

Table 4: Evaluation of validation perplexity when number of 'thinkings tokens' differs

We have also investigated how the number of 'thinking tokens' influences the result as shown in Table 4.

Adding more 'thinking tokens' worsens the validation perplexity with LSTM model. Explanation of this trend could be that using more 'thinking tokens' is not always beneficial, since it increases the chance of a model to forget what was before 'thinking tokens'.

A.5 Model hyper-parameters

Hyper-parameters	
Parameter	**Value**
Bptt	70
Batch size	12
Gradient clipping	0.25
Hidden neurons	450
Layers	1
Hidden neurons	450

Table 5: Model hyper-parameters

Hyper-parameters used to train our LSTM are listed in Table 5. We have used the ASGD trick in training [12].