,

A combination of autoregressors and autoencoders using XLNet for sentiment analysis

Charalampos Angelakopoulos CH.ANGELAKOPOULOS@GMAIL.COM
Master's student
National Technical University of Athens
Athens, Greece

Dimitrios Zaridis DIMZARIDIS@GMAIL.COM
PhD student
National Technical University of Athens
Athens, Greece

Dimitrios Oikonomou DIMITRIS.OIK96@GMAIL.COM
Master's student
National Technical University of Athens
Athens, Greece

Editor: Charalampos Angelakopoulos, Dimitrios Zaridis, Dimitrios Oikonomou

Abstract

In this paper sentiment analysis has been performed in order to evaluate the performance of XLNet on this particular task. XLNet is rather a ground-breaking network on language understanding which uses the perks of both autoregressive models and autoencoders. While BERT uses autoencoders and Transformers use autoregression, XLNet combines the aforementioned networks' attributes in order to achieve higher performance in many NLP tasks, such as sentiment analysis, question answering, reading comprehension, natural language understanding etc. In this work we evaluate the XLNet model in several sentiment classification tasks in terms of accuracy and efficiency. The XLNet reaches state of the art results and outperforms BERT which is the previous state of the art model on natural language processing.

Keywords: Deep learning, Natural Language Processing, Sentiment Analysis

1. Introduction

Natural Language Processing (NLP) refers to the branch of computer science and more specifically the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to "understand" its full meaning, complete with the speaker or writer's intent and sentiment. NLP tasks could be many in numbers and some examples of those tasks are presented below.

- Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar. Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context.
- Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb make in "make the grade" (achieve) vs. "make a bet" (place).
- Named entity recognition distinguishes words or entities that they have a particular meaning such as an attribute. For instance Mississippi is a river etc.
- Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers but it can also involve identifying a metaphor or an idiom in the text.
- Sentiment analysis attempts to extract subjective qualities, attitudes, emotions, sarcasm, confusion and suspicion from text. (Zhang et al., 2015)
- Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There's a good chance someone has interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes. The understanding of natural language is rather a challenging task for machines to perform due to the fact that natural language has only meaning for human recognition and therefore machines need to learn more robust and abstract features from text.

2. Related work

Several studies have been performed in sentiment analysis tasks which either use autoregressive models or autoencoders (Germain et al., 2015). The methods of presenting and inducing the tokens are extremely effective due to the unsupervised way of training which do not require annotation by human and therefore can represent the words in terms of syntactic and semantic context. Several huge steps have been made in pretrained networks and one of them is the BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) and that architecture is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers and as a result the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art results for a wide range of tasks, such as question answering and language inference. The main disadvantage of BERT though is that it assumes that tokens, which are actually the words in a text corpus, are independent of each other but this assumption is actually wrong due to the fact that in natural language, phrases and sentences are meaningful and one can infer the basic meaning of a sentence just from reading some first words. On that direction XLNet (Yang et al., 2019) was proposed in order to tackle the cons of BERT. Specifically, the main advantage of the XLNet is that it maximizes the expected log likelihood of a sequence in a manner that all possible permutations of the factorization (Ada) are used for context extraction. From this operation, each position of the token can learn to utilize contextual information from all the remaining positions. Furthermore, the utilization of autoregressive language model is capable to use the product rule to factorize the joint probability of the predicted tokens and therefore the independence assumption is eliminated. Moreover, a path towards explainability is crucial in natural language interpretation. For language understanding, interpretability is very important for the model in order to provide more robust and reliable results. Especially, self explaining structures (Sun et al., 2020) have been introduced in order to improve the performance of NLP models. In the aforementioned work, the proposed method is to put an additional layer, as is called by the interpretation layer, on top of any existing NLP model and provides the following perks.

- Span weights make the model self-explainable and do not require an additional probing model for interpretation.

- The model can be easily generalized and used in any NLP task.

- More emphasis is given into the phrases and sentences via weighting those tokens.

2.1 BERT

BERT is an autoencoder language model that does not need separate position information like an autoregressive language model. Unlike the XLNet which needs position information to predict the t-th token in a sentence, BERT uses a [MASK] token to represent which token to predict (we can think [MASK] just as a placeholder). For example, if BERT uses tokens x_1, x_2 and x_4 to predict token x_3 in a sentence then the tokens x_1, x_2 and x_4 contain the position information and other information related to [MASK]. So the model has a high chance to predict that [MASK] corresponds to token x_3. (Liu et al., 2019). Here a more detail explanation is given about information. BERT embedding (information learned by BERT) contains two kinds of information, the position information and the content information as can be seen in Figure 1.

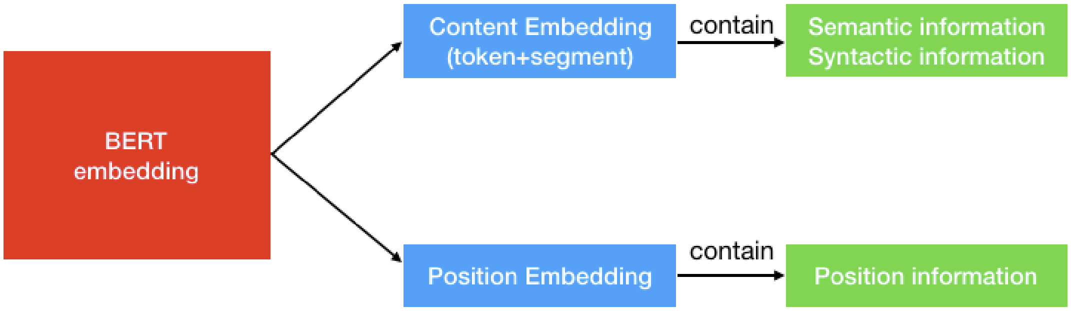


Figure 1: BERT's workflow (Horev, 2018).

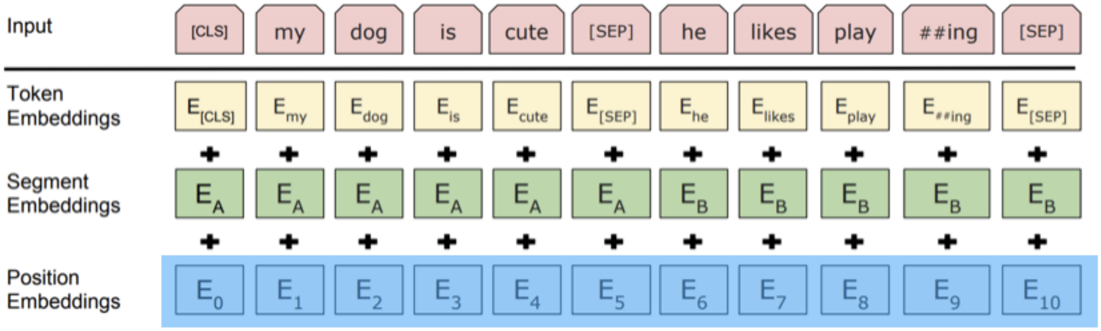


Figure 2: BERT'S encodings.

The position information is easy to understand because it just tells where is the position of the current token. The content information (semantics and syntactic) contains the "meaning" of the current token.

3. Methods

3.1 Intuition behind XLNet

XLNet proposes the two-stream self-attention and as the name indicates it contains two kinds of self-attention. The first ont is the content stream attention which is the standard self-attention in a Transformer architecture. The other one is the query stream attention. XLNet introduces this idea in order to replace the [MASK] token in BERT. For example, if BERT wants to predict token x_3 with knowledge of the context words x_1 and x_2, it can use [MASK] to represent the x_3 token. The embeddings of x_1 and x_2 contain the position information to help the model conclude that [MASK] corresponds to token x_3. Things are different with XLNet model. The token x_3 will serve two kinds of roles. When it is used as content to predict other tokens, we can use the content representation (learned by content stream attention) to represent x_3. But if we want to predict x_3, we should only know its position and not its content. That's why XLNet uses query representation (learned by query stream attention) to preserve context information before x_3 and only the position information of x_3 (Figures 3, 4).

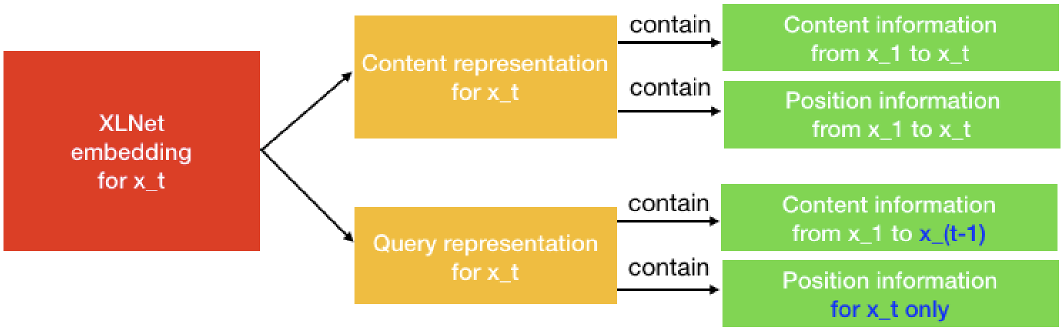

XLNet embedding for x_t
Content representation for x_t
Query representation for x_t
contain
contain
contain
contain
Content information from x_1 to x_t
Position information from x_1 to x_t
Content information from x_1 to x_(t-1)
Position information for x_t only

Figure 3: XLNet's encodings (Jagtap, 2020).

In order to intuitively understand the Two-Stream Self-Attention, we can just think XLNet replaces the [MASK] in BERT with query representation. They just choose different approaches to do the same thing. Now we give a brief explanation of the implementation of the XLNet model. Let's say a sentence has 4 tokens $[x_1, x_2, x_3, x_4]$. It is very easy to misunderstand that we need to get the random order of a sentence and input it into the model. But this is not true. The order of input sentence is $[x_1, x_2, x_3, x_4]$ and XLNet uses attention mask to permute the factorization order. We randomly get a factorization order such as $[x_3, x_2, x_4, x_1]$. The upper left corner is the calculation of content representation. If we want to predict the content representation of x_1 we should have all 4 tokens content information.

3.2 Mathematical expressions

XLNet could be described as the continuation of transformers, for transformers can rely on two attributes

- to predict the token x_t, the model should only see the position of x_t, not the content of x_t
- to predict the token x_t, the model should encode all tokens before x_t as the content

While the regular Transformer-XL (Dai et al., 2019) parametrization may not work with the permutation language model. To understand this, let's consider the standard formulation

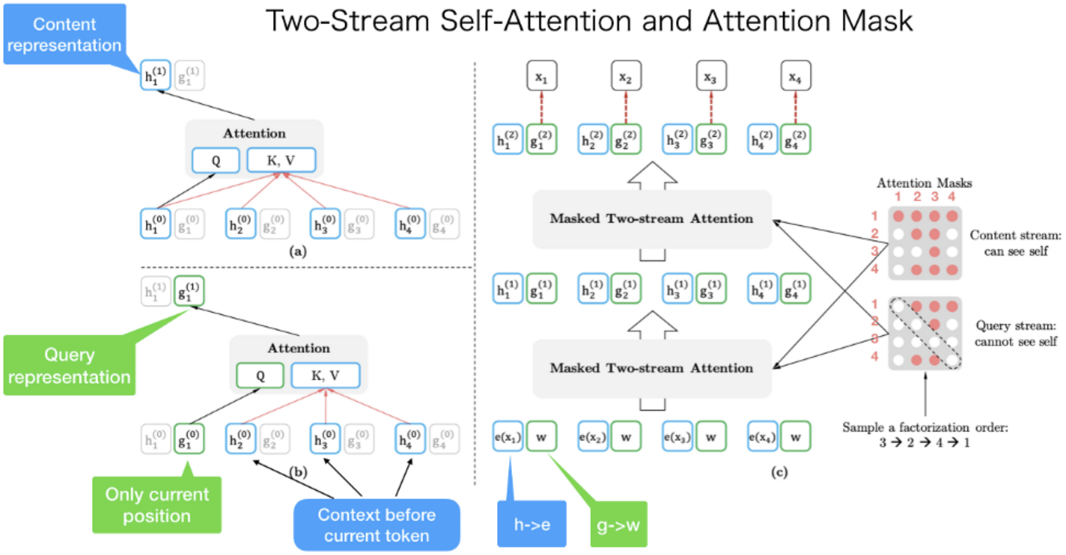

Two-Stream Self-Attention and Attention Mask
Content representation
$h_1^{(1)}$ $g_1^{(1)}$
Attention
Q
K, V
$h_1^{(0)}$ $g_1^{(0)}$ $h_2^{(0)}$ $g_2^{(0)}$ $h_3^{(0)}$ $g_3^{(0)}$ $h_4^{(0)}$ $g_4^{(0)}$
(a)
Query representation
(b)
Only current position
Context before current token
x_1 x_2 x_3 x_4
$h_1^{(2)}$ $g_1^{(2)}$ $h_2^{(2)}$ $g_2^{(2)}$ $h_3^{(2)}$ $g_3^{(2)}$ $h_4^{(2)}$ $g_4^{(2)}$
Masked Two-stream Attention
$h_1^{(1)}$ $g_1^{(1)}$ $h_2^{(1)}$ $g_2^{(1)}$ $h_3^{(1)}$ $g_3^{(1)}$ $h_4^{(1)}$ $g_4^{(1)}$
Masked Two-stream Attention
$e(x_1)$ w $e(x_2)$ w $e(x_3)$ w $e(x_4)$ w
(c)
h->e
g->w
Attention Masks
1 2 3 4
Content stream: can see self
Query stream: cannot see self
Sample a factorization order: 3 → 2 → 4 → 1

Figure 4: 2-stream self attention model (Jagtap, 2020)

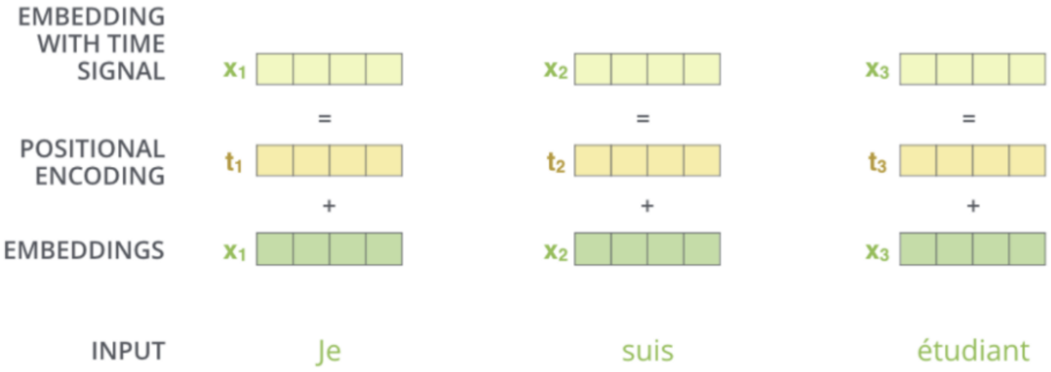


Figure 5: Encoding paradigm (Jagtap, 2020)

of the distribution using softmax which is given by:

$$softmax_{transformer} \approx \frac{e^{e_x^T} h_\theta x_{z_{<t}}}{\Sigma_{x'} e^{e^{x'T}} h_\theta x_{z_{<t}}} \tag{1}$$

Here, the term $h_{(x_{z_{<t}}}$, is the hidden state of the transformer for $x_{z_{<t}}$. This term, is in no way dependent on the position that it predicts i.e. $z_{<t}$. This means, that whatever position is being predicted, this distribution will be the same; thus posing the inability to learn useful trends.

Some ideas already exists in regards of permutation modeling such as B.Uria's work (Uria et al., 2016) and in XL-net some basic concept of the caption of bidirectional concepts are retained. Specifically, for a sequence x of length T, there are T! different orders to perform a valid autoregressive factorization. Intuitively, if model parameters are shared across all factorization orders, in expectation, the model will learn to gather information from all positions on both sides. To formalize the idea, let Z_T be the set of all possible permutations of the length-T index sequence $[1, 2, \ldots, T]$. We use z_t and $z < t$ to denote the t-th element and the first t1 elements of a permutation $z \in Z_T$. Then, the XLNet' s permutation language modeling objective can be expressed as follows:

$$\max_{\theta} \mathbb{E}_{\mathbf{z}\sim Z_t}\Big[\sum_{t=1}^{T} \log p_{\theta}(x_{z_t}|\mathbf{x}_{\mathbf{z}<t})\Big] \tag{2}$$

Essentially, for a text sequence $\mathbf{x}$, we sample a factorization order z at a time and decompose the likelihood $p_{(x)}$ according to factorization order. Since the same model parameter is shared across all factorization orders during training, in expectation, x_t has seen every possible element $xi \notin xt$ the sequence, hence being able to capture the bidirectional context. Moreover, as this objective fits into the AR framework, it naturally avoids the independence assumption. Hence, the XL-net paper proposes a re-parametrization for the next token distribution to be target aware:

$$softmax_{XLNet} \approx \frac{\exp(e_x^T h_\theta g_\theta(x_{z_{<t}}, z_t))}{\Sigma_{x'}\exp(e_x^T h_\theta g_\theta(x_{z_{<t}}, z_t))} \tag{3}$$

A modified representation g is used, which additionally takes the target position z_t as the input. So, two hidden states are used instead of one. The content representation, which is essentially the same as the standard Transformer hidden state. This representation encodes both; the context $x_{z_{<t}}$ as well as the original token x_{z_t}.

- The content representation, which is essentially the same as the standard Transformer hidden state (Vaswani et al., 2017). This representation encodes both: the context $x_{z_{<t}}$ as well as the original token x_{z_t}. Mathematically this can be notated as follows:

$$h_{z_t}^{(m)} \leftarrow Attention(Q = h_{z_t}^{(m-1)}, KV = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta) \tag{4}$$

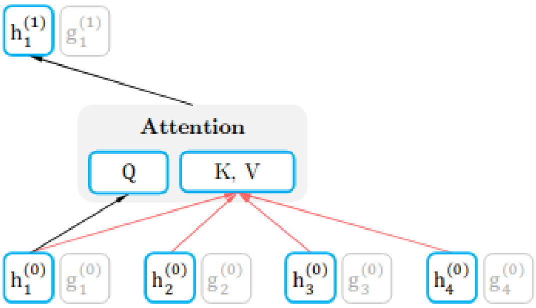


Figure 6: Content stream attention (Liang, 2019a)

- The query representation, that has access only to contextual information $x_{z_{<t}}$ and the position of the target z_t.

$$g_{z_t}^{(m)} \leftarrow Attention(Q = g_{z_t}^{(m-1)}, KV = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta) \tag{5}$$

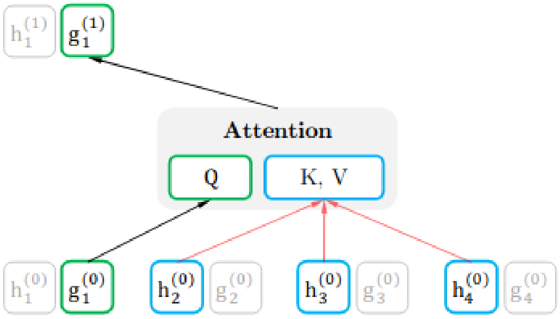


Figure 7: Query stream attention (Liang, 2019a)

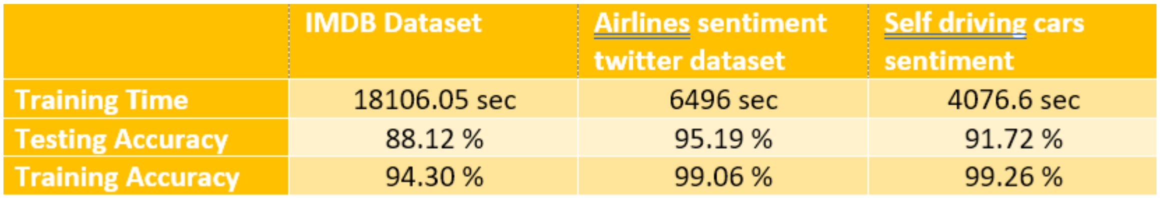
A useful notation here is that initially the content stream h_i is essentially the corresponding embedding vector e_{xi}, and the query stream g_i is a trainable vector w initially. These are updated over each layer using the above expressions.

3.3 Datasets

In this work, we build a sentiment classifier using the XLNet model, which evaluates the polarity of a piece of text being either positive or negative. The first dataset we used is the IMDB dataset (Zhu et al., 2015) and contains the "Large Movie Review Dataset". The dataset consists of 50,000 reviews from IMDB on the condition that there are no more than 30 reviews per movie. The number of positive and negative reviews are equal. Negative reviews have scores less or equal than 4 out of 10 while a positive review have score greater or equal than 7 out of 10. Neutral reviews are not included. The 50,000 reviews are divided into the training (25,000 reviews) and test set (25,000 reviews). This could be referred as a text or sentiment classification task. The second dataset that we used is the twitter Airline sentiment which includes reviews about six american airline companies together with the reason of a negative review. In the present work we keep only the classification of the review (positive or negative) and the text of the review in order to train and evaluate the XLNet model. The amount of data were approximately 12.000 tweets. Specifically 9232 tweets were used for training and 2309 for test. The third dataset is the self-driving sentiment analysis from tweeter users. Users opinions about self driving cars are included in the dataset in terms of zero for negative ones and 1 for positive ones. Here the training was partioned with 8900 opinions and the test set with 3500 opininions.

4. Results

Our results are presented below in table.1



	IMDB Dataset	Airlines sentiment twitter dataset	Self driving cars sentiment
Training Time	18106.05 sec	6496 sec	4076.6 sec
Testing Accuracy	88.12 %	95.19 %	91.72 %
Training Accuracy	94.30 %	99.06 %	99.26 %

Figure 8: Results of XL-net for 3 different datasets

5. Discussion and Conclusions

Keeping aside all the benefits that permutation Language Modelling has, we need to accept that it is expensive. It is a challenging optimization problem due to permutation. Hence, to solve this, in a given sequence z, only a subsequence $z_{>c}$ is selected for prediction, where c is called the cutting point. We consider only $z_{>c}$ since it has the longest context in that sequence. Moreover, another hyperparameter K is used such that, $K \backsim \frac{|z|}{(|z| - c)}$. And we select only 1/K tokens for prediction. For unselected tokens, their query representations are not computed, which saves speed and memory. We compare this partial prediction to that of the BERT. BERT uses partial prediction because masking all the tokens does not make any sense. XLNet does partial prediction because of the optimization difficulty. For example: let's have a sequence: [Car, needs, fuel, to, start]. Say both BERT and XLNet opt to predict the tokens [Car, needs]. Also suppose that XLNet factorizes the sample as [fuel, to, start, Car, fuel]. In this case BERT maximizes:
$\mathrm{L}_{BERT} = \log p(Car|fuel\ to\ start) + \log p(needs|fuel\ to\ start)$, while XLNet maximizes:
$\mathrm{L}_{XLNet} = \log p(Car|fuel\ to\ start) + \log p(Car|needs\ fuel\ to\ start)$.

This clearly explains how XLNet captures more dependency i.e. between Car and needs. No doubt that BERT learns most of the dependencies but XLNet learns more (Liang, 2019b). Also, this is an example of the independence assumption in BERT which was covered in the previous section.

Acknowledgements

We would like to acknowledge support for this project from the Deep learning course of the Artificial Intelligence and Learning Systems Laboratory (AILS), National Technical University of Athens, Greece.

References

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. URL `http://arxiv.org/abs/1901.02860`.

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. URL `http://arxiv.org/abs/1810.04805`.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/germain15.html`.

Rani Horev. Bert explained: State of the art language model for nlp, 2018. URL `https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270`.

Rohan Jagtap. Xlnet: Autoregressive pre-training for language understanding, 2020. URL `https://towardsdatascience.com/xlnet-autoregressive-pre-training-for-language-understanding-7ea4e0649710`.

Xiu Liang. What is two-stream self-attention in xlnet, 2019a.

Xiu Liang. What is xlnet and why it outperforms bert, 2019b. URL `https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335`.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. Self-explaining structures improve NLP models. *CoRR*, abs/2012.01786, 2020. URL `https://arxiv.org/abs/2012.01786`.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *CoRR*, abs/1605.02226, 2016. URL `http://arxiv.org/abs/1605.02226`.

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. URL `http://arxiv.org/abs/1706.03762`.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019. URL `http://arxiv.org/abs/1906.08237`.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015. URL `http://arxiv.org/abs/1509.01626`.

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. URL `http://arxiv.org/abs/1506.06724`.