XLNet: Generalized Autoregresive Pretraining for Language Understanding Summary

Blake Hillier
Advanced Big Data Analysis

February 10, 2020

1 Background

This is a summary of the paper *XLNet: Generalized Autoregresive Pretraining for Language Understanding* by Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Neural networks are commonly used when dealing with text, ranging from sentiment analysis to text creation. These networks are often pretrained on massive amount of texts before being finetuned to the problem at hand in order to give the model a basic understanding of language.

1.1 BERT

One of the most efficient pretraining methods is BERT, an autoencoding method. Given a sequence of words, it randomly hides some words to learn through sentence reconstruction. Mathematically the model solves this equation:

$$\max_{\theta} \ \log p_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{x}) = \sum_{t=1}^{T} m_t \log \frac{e^{H_\theta(\hat{x})_t^T l(x_t)}}{\sum_{x'} e^{H_\theta(\hat{x})_t^T l(x')}}$$

where x_t is the tth word, $m_t = 1$ if x_t is hidden and 0 otherwise, H_θ turns a sequence of text x into a sequence of hidden words, and $l(x)$ is the embedding of x. This allows it to understand relationships between words in a section of text. However, it assumes each hidden word is unrelated, limiting it's understanding to local sections instead of large sections. It also creates an imbalanced distribution when finetuned. These problems are avoided when using a different group of methods called autoregressive models (AR)

1.2 AR

Autoregressive models create a conditional probability distribution based on text during pretraining. They calculate a likelyhood function

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{<t}) \ \text{ or } \ p(x) = \prod_{t=T}^{1} p(x_t|x_{>t})$$

where $x_{<t}$ is the sequence of text from 1 to $t-1$, and $x_{>t}$ is sequenced from $t+1$ to T. This allows the model to understand the forward or backwards relationship between text, but not both. The paper attempts to combine both methods as a way to merge the benefits while claiming the negatives disappear.

2 XLNet

The XLNet model attempts to maximize

$$\max_{\theta} \ \mathbb{E}_{z\sim Z_T}\left[\sum_{t=1}^{T} \log p_\theta(x_{z_t}|x_{z<t})\right] = \mathbb{E}_{z\sim Z_T}\left[\sum_{t=1}^{T} \log \frac{e^{g_\theta(x_{z<t,z_t})l(x_t)}}{\sum_{x'} e^{g_\theta(x_{z<t,z_t})l(x')}}\right]$$

where Z_T is the set of all permutations of text of length T, $z \in Z_T$, $x_{z<t}$ is the sequence of text from 1 to $t-1$, and g_θ transforms x to a sequence of hidden words with the first $t-1$ set of words as additional information. They note the permutations don't affect the actual order of the text sequence, just the factorization of the likelyhood function. Because this is based on the likelyhood function, it removes the idea each hidden text is independent of the others while still maintaining the benefits through encoding due to g_θ. In order for g_θ to accomplish this, they split it into two different transforms: the query g_θ which looks at the first $t-1$ words in the permuted order to predict the t^{th} word, and the content h_θ which simply encodes the first t words in the permuted order. The one downside to this model is the complexity of the optimization leading to a slower convergence. To reduce this problem, they adjust the equation to

$$\max_\theta \ \mathbb{E}_{z\sim Z_T}\left[\log p_\theta(x_{z_{>c}}|x_{z_{\leq t}})\right] = \mathbb{E}_{z\sim Z_T}\left[\sum_{t=c+1}^{|z|} \log p_\theta(x_{z_t}|x_{z_{<t}})\right]$$

which changes the model to only predict the last $T-c$ words in the permutation order, where $c \approx \frac{T(1+K)}{K}$ for some parameter K.

3 Experiments

They used 32.98B of data after filtering and tokenizing the data. Both BERT and XLNet used 512 tokens per sequence, used adam as the optimizer, and pretrained each model for about 5.5 days. They then used multiple natural language understanding datasets (such as SQuAD or RACE) to judge the ability of each model. First they used the same hyperparameters and data for both models and found XLNet significantly outperformed BERT. XLNet was then tested against a newer model, RoBERTa, using the same method as with BERT. XLNet still outperformed RoBERTa significantly when context was needed, and still performed better with classification tasks.

4 Conclusion

The results from the last section show their pretraining model is better than some of the best models currently out there. This benefits our class project due to our need for sentiment analysis, which depends on an NLP model's ability to understand context. Training our sentiment analysis model with this pretraining method will result in a more accurate model, boosting the overal ability of the rest of our trading strategy.