Pre-trained Language Model: Survey

Wong Tsz Ho
Student ID: 20725187
thwongbi@connect.ust.hk

Abstract

The survey aims to examine pre-trained language models. We will cover everything from defining a model to the evolution of it. Various popular pretrained language models with techniques and methodologies will be covered, including BERT, GPT, T5, OPT, and LaMDA. A brief review of recent developments in benchmarking will also be provided. Several Python libraries will be presented, including NLTK, SpaCy, HuggingFace, TensorFlow, and PyTorch. We will conclude our survey with a discussion on the application of the pre-trained language models.

1 Introduction

Among the hottest topics in NLP, pre-trained language models have gained popularity. The model opened up the possibility of a generic and universal language model that solves downstream language problems and functions as a knowledge base. This is a significant milestone in the creation of Strong AI.

In this survey, I will discuss the evolution of the pre-trained model from the past until now, and introduce the most popular pre-trained language model using benchmarking. Following that, I will discuss the existing tools for applying those language models, and the survey will conclude with some applications of those pre-trained language models. By putting this information into context, audiences will be able to gain a better understanding of the current development of pre-trained language models, allowing them to step into this exciting area with ease.

2 Language Model

A language model is just a probability distribution of the sequence of words. Simply saying, it is the probability of saying that the word sequence is making sense or not. In most cases, the input of a language model would be a sentence, and the language model will determine whether or not the sentence is meaningful.

The reason why a language model is so important to neutral language processing is that the language model gives meaning to the mathematical model and the machine learning model can now take the language sense into account when they made the decision.

The recent development of language models would be on masked language models. Instead of predicting the next word, masked language models can handle masked words in the middle of the word sequence. BERT would be one of the masked language models.

3 Language Model Taxonomy

3.1 Statistical Language Models

3.1.1 N-gram Language Models

Way before the rise of the neural network, scientists find some ways to represent a language with n-gram. N-gram is a moving N window of a word sequence and we use the distribution of the combination of n-grams to predict the next word sequence.

A unigram language model does not take the sequence of the word into its model and only generates the word sequence using the distribution over the vocabulary. While bigram model takes two words into account and calculates the probability distribution over the corpus. Using a high value in N can improve the meaning of the generated word sequence but comes at the cost of computation and storage problems.

3.2 Neural Network Language Model

Recurrent Neural Network like LSTM[1] introduces sequencing with the input element where language by nature has order. RNN also allows the model weight to be constant with respect to any length of the input because the model ingests input one by one.

In 2003, Bengio[2] proposed that word embedding can be learned using a neural network. Since then the research on neural Network Language Model boomed. The concept of using words as vectors create many possibilities for treating words. Pretrained word2vec[3] model, from Google, then appears in the market and caused a huge impact on the NLP community. The benefit of expressing the words as vectors is that you can treat those words as numbers and are able to do mathematical operations like addition, subtraction, etc. Something like, "girl"+"women"-"men" = "boy" would make sense in vector space. Word2Vec is not the only pre-trained word embedding, GloVe[4], from Stanford University, is also the other popular word representation. This difference between the two models is that Glove would consider the co-occurrences information. FastText[5], from Facebook, further takes unknown words from generalization.

4 Pre-trained Language Model

A pre-trained language model nowadays usually is trained on a large corpus where the model can represent the universal language. Using those pre-trained language model can save us times and effort to work on everything from scratch. Several hundred models parameters have been added to deep learning models since the advent of deep learning.

For full parameter training and to prevent overfitting, a much larger dataset is required. The high annotation costs for NLP tasks, especially for the semantically and syntax-related tasks, make constructing large-scale labeled datasets a challenge for most NLP tasks. Pre-trained language model is indeed helping the evolution of humanity to accelerate faster than ever before. To see how far we have come, I will recount a few of the pre-trained language models that we have developed.

4.1 BERT

In 2018, BERT [6]or Bidirectional Encoder Representation from Transformers was born in Google. It is a Transformer Language Model. It consists of self-attention heads and multiple encoder layers. With its bidirectional nature, it does not encode the same word into one vector, instead, it will consider the semantics of the sentence.

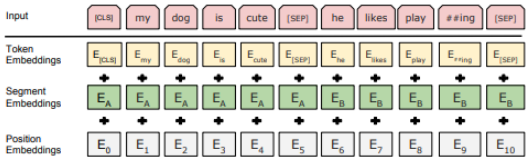

Figure 1: BERT[6]: Input Representation

BERT model is pre-trained together with the masked language model and next sentence prediction. This allows the combined loss function would be as low as possible. As mentioned earlier in the survey, a masked language model is the language model that will randomly mask out the words as an input to the model. For BERT, 15% of the words are masked and the masked word is replaced by a [MASK] token. Next sentence prediction from BERT will take two sentences as input and learn whether or not they are the next sentence. Combining these two training will result in a lower loss of the entire model. BERT model is available on HuggingFace.

4.2 GPT

GPT stands for Generative Pre-trained Transformer, and the latest version is GPT-3[7]. It is developed by OpenAI. GPT-3 trained on 175B parameters with open-source dataset named 'Common Crawl' which has around 45TB of text. GPT-3 is not open source like GPT[8] and GPT-2[9] and hence you cannot find GPT-3 online. Pre-trained GPT-1/GPT-2 models are in HuggingFace and you can directly implement the model using the Python package. There is one benefit of using GPT-3 is that it does not require developers to fine-tune to perform tasks.

4.3 T5

Text-To-Text Transfer Transformer[10], also known as T5, suggest that every task we would like to do is an question and answer pairs. Tasks like translation, classification, chat bot, would input as text to the T5 model and the T5 model will

generate the target text. For example, to ask T5 to translate "This is awesome." to Chinese, your input to the T5 model would be "translate English to Chinese: This is awesome.". You can find the T5 model on HuggingFace.

4.4 OPT

Meta AI proposed the OPT model in the paper Open Pre-trained Transformer Language Models[11]. OPT is a family of large, open-sourced causal language models that perform similarly to GPT3. A difference between GPT2 and OPT is that OPT adds the EOS token </s> at the beginning of every prompt. Same as other transformers model, you can find the OPT model on HuggingFace.

4.5 LaMDA

LaMDA[12], Language Model for Dialogue Applications, is a generative language model by Google. It is also built on Transform. The special thing about this model is that it is trained on the dialog rather than the web text or wiki page like in GPT. It gives that LaMDA model is for Google to improve their Google assistant. And they just announced that they had released the next version LaMDA in Google I/O 2022, which is built on a new Pathways[13] system which allows the model to scale to 540B parameters.

5 Benchmarking a Language Model

Benchmarking is important for a researcher to evaluate how well their models are. GLUE[14], General Language Understanding Evaluation benchmark, is one of the popular benchmarking metrics. It provides tools for evaluating and analyzing the performance of models of natural language understanding across a range of existing tasks proposed as part of the GLUE. With the rapid development of the NLP models, most of the models can score really high in their dataset and benchmark, and hence, two years after the launch of GLUE, they launch the successor of this benchmark, which is called SuperGLUE[15].

6 Language Model with Python

6.1 Language Model library

6.1.1 NLTK

NLTK[16] is the Natural Language Toolkit in Python. It is free and open-source. It provides useful utility to work on word with Python. With NLTK, you can tokenize and tag text with ease and hence you can identify named entities. It embedded datasets where you can fine-tune your custom model with NLTK. NLTK comes with a package called NLTK.LM where it currently only supports n-gram language models. NLTK does not provide a neural network language model for more advanced use.

6.1.2 SpaCy

SpaCy[17], when compare to NLTK, can process NLP task faster. It provides pre-trained models and a pipeline for building your NLP application. There are 4 pre-trained pipelines for English, *en_core_web_sm*, *en_core_web_md*, *en_core_web_lg* and *en_core_web_trf*. The name represented here was that it is an English general language model trained from Web text which includes blogs, and news. The last naming represents the size of the model while trf represents that it is a transformer model roberta-base. Spacy package is optimized for CPU running and hence it is suitable for fast inference of model to give an immediate response on the lightweight application. Another special thing spacy provides is that it has a Chinese pre-trained model free to use.

In our project, Toxic language detection/ debiasing toxic content, this is the one we use to quickly build the baseline of the task. We tagged the toxic language words using IOB[18] format which can give us a fair result on the entity recognition task. After we tagged all of the toxic text from the dataset, we bind the IOB file to the pre-trained SpaCy English model. After hours of fine-tuning the pre-trained model, the pipeline is ready to use. The text will first transform to a vector using tok2vec. Then the tagger files will be loaded and all our toxic text will be bound with the vector and the Named Entity Recognition will identify toxic text from the comments.

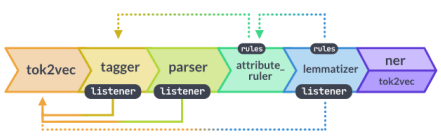

Figure 2: SpaCy[17] Pre-trained Pipeline Architecture

6.1.3 HuggingFace

HuggingFace[19] rise after the transformer revolution that started in 2016. It is a repository of community-based NLP pre-trained language models. It serves a number of transformer model by tasks that you can easily select the right model for your problem. There are nearly 46k pre-trained models and of course they have the popular BERT and GPT2 .

In our project, Toxic language detection/ de-biasing toxic content, we utilize HuggingFace provided BERT/DistillBERT model to fine-tune the pre-trained model. Using HuggingFace's pre-trained model saves us time on training the model from scratch. The BERT model has 12 attention layers and all these layers combined has 110 million trainable parameters while 66 million for DistillBERT. Unlike Word2Vec, the tokenizer in BERT/DistillBERT splits tokens into subtokens on less commonly seen word which it helps the model to generate and handle an unseen word.

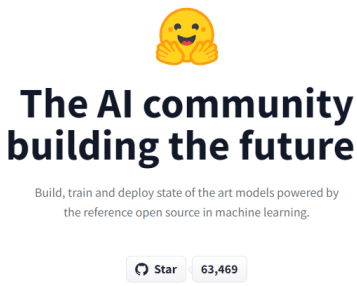

Figure 3: HuggingFace[19]: Logo and GitHub Stars

6.1.4 TensorFlow & PyTorch

TensorFlow[20] and PyTorch[21] are two popular generic deep learning library. While you can train your own language model, both of them provide datasets and a pre-trained language model for us to use. TensorFlow has set up a Tensor Hub for the community to host their models on TensorFlow. Not only can you find Language models there, you can find other pre-trained models in Audio, Video, and Image as well. PyTorch Hub is the place where PyTorch stores its pre-trained model for a developer to use. You can find pre-trained language models like GPT-2, and BERT in PyTorch Hub.

6.2 Other NLP Python Library

There are a few more python packages worth mentioning. By Humboldt University Berlin, Flair[22] is a framework designed for developing state-of-the-art natural language processing. It allows us to perform text embedding with simple interfaces on Python. This package is built on top of PyTorch and hence it is fully extensible to other PyTorch applications.

Gensim[23] provides ready-to-use corpora and models with a streaming algorithm to load and process NLP tasks on the go without loading all data onto your memory. You can easily load and use pre-trained model like word2vec[3] and FastText[5] for pre-processing or other NLP Task.

6.3 Application of Pre-trained LM

Apart from what Language Model is built for, predicting the next word and calculating the probability that a sentence makes sense or not, there are a number of interesting tasks that a pre-trained language model can help with. NLP tasks like text categorization, speech recognition, Neural Machine Translation, and information retrieval can all boost the performance with a pre-trained language model. This language model can then be used to perform NLP tasks in different domains.

Since a pre-trained language model has a feed with lots of real-world knowledge on the web, it makes sense that sometime the next word prediction can treat it as knowledge for answering the prompt. Like when you ask "Albert Einstein was born in ___", and the next word that the language model predicts should be the exact date he was born. Because most of the text we feed into the model is some sort of fact in the world and hence we can utilize a pre-trained language model as a knowledge base. Acquiring enough commonsense knowledge, we are one step closer to developing a generic Strong artificial intelligence (AI)[24].

Pre-trained Language Model is still an active growing study area and it is so exciting to see how the community and the model evolve over the year hence this survey is research that helps you get into this area and admire how far we have been in this area.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. ISSN 1532-4435.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016. URL https://arxiv.org/abs/1607.04606.

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

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

[8] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[11] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

[12] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[13] 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.

[14] Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018. URL http://arxiv.org/abs/1804.07461. cite arxiv:1804.07461Comment: https://gluebenchmark.com/.

[15] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

[16] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python.* O'Reilly Media Inc., 2009.

[17] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[18] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040, 1995. URL http://arxiv.org/abs/cmp-lg/9505040.

[19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[22] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[23] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[24] Martin V Butz. Towards strong ai. *KI-Künstliche Intelligenz*, 35(1):91–101, 2021.