What does Pre-training Help in Language Understanding?

Research Proposal

Xuhui Zhou
Department of Mathematics
Nanjing University
Nanjing, JiangSu

December 8, 2018

Abstract

The pre-training of language models on larger unlabeled dataset has substantially improved many natural language processing (NLP) tasks such as dependency parsing, named entity recognition, and textual entailment. Natural language understanding (NLU), as a term associates with question answering, commonsense reasoning, and semantic similarity assessment, also benefits from the development of pre-training techniques. Recently, the emergence of deep contextualized pre-training, which can leverage more than word-level information, has pushed state of the art of many NLU tasks to a whole new level and gained much attention from researchers and public subsequently. In particular, the BERT[1], which is language representations from transformers, has improved the commonsense inference task (SWAG) to around the human levels (88.0%), while many current natural language inference (NLI) models stay around 60%. In this work, we investigate the empirical reason of why pre-training, especially contextualized pre-training, is so powerful in NLU, specifically in commonsense inference. In contrast to previous work[2] discussing how pre-training works in deep learning generally, we focus on explaining the roles of pre-training in language understanding compared to humans' massive reading process. We hypotheses that pre-training is an effective way to let our trained language model approximate humans' language model and expect this hypothesis to serve as an evidence that pre-training can make our models "think" the same as humans despite the various defects in our current NLI datasets[3]. We further propose two novel metrics *human mean factor* and *human variance factor* to quantify the distance between trained and humans' language models. We hope this two metrics could serve as a guide for future development of algorithms. We also train different pre-training with various objectives and test them in SWAG with different amounts of data.

1 Introduction

Unsupervised pre-training, as a way of utilizing abundant unlabeled data, has brought substantial improvement to the deep learning community such as computer vision, speech recognition[4], and NLP. By learning effectively from abundant raw unlabeled texts, models generalize better in many domains that suffer from a dearth of labeled resources and get better performance even in cases where considerable labeled data is available. The tasks that benefit from pre-training include part-of-speech tagging[5], named entity recognition[5], dependency parsing[6], sentence classification[7], and machine translation[8]. With such a broad applications of pre-training in NLP, however, there are few works discussing the way it works such as [8] and [2]. While the aforementioned pre-training techniques, specifically word embeddings, are leveraging the word-level information, recent research such as [9] and [1] made successful attempts at utilizing contextualized information from the unlabeled text corpora and further improved most of NLP tasks. However, we still have little knowledge about the exact roles that pre-training played in many NLP tasks, especially those tasks that requires more than word-level representations.

He is up a tree. Someone	
a) stands underneath the tree. (97.44%)	likely, best
b) is at a pool table holding a cup. (1.14%)	likely
c) grabs a flower from a paper. (0.96%)	unlikely
d) is eating some cereal. (0.45%)	likely

Table 1: Example question answered by ELMo model, model probability is in the parentheses, right side is the expert judgment.

Language understanding is a sophisticated and subtle task which has drawed lots of researchers' atenttion recently, partly because of the introduction of SWAG[3]. SWAG is a new dataset with questions about grounded situations (See 1), which addresses the annotation artifacts (i.e. conditional stylic patterns) and human biases with adversarial algorithms. Previous human-written NLI datasets are susceptible to introduce "gamed" patterns, (e.g. annotation artifacts), which have been reported to allow models like bag-of-word reach artificial high performance[10][11]. SWAG, however, successfully reduces the biases and makes many state of the art NLP algorithms well below the human-level[3] (88.0%).

Algorithms with various pre-training techniques have always been the first on the leader board ever since the emergence of SWAG. At first, ELMo[9], which is a representative of feature-based pre-training approach, reaches around 60%. Then, GPT[12] uses fine-tuning approach, which is trained on the downstream tasks by simply changing its parameters, get 77.97%. Now, the BERT[1] demonstrates the power of the pre-trained representations by modifying standard transformer models to bidirectional models and answers the questions in SWAG just as well as humans did. While there is a difference between algorithms with and without pre-training, there are substantial differences among different methods of pre-training in such a task heavily requires commonsense reasoning. We are not only interested in what pre-training helps in this process, but also why some pre-training methods help more.

Previous work[2] shows that pre-training is equivalently a regularizer from the perspective of optimization .We argue that this is not a satisfactory explanation for improving language understanding tasks. Instead, we compare the process of pre-training to humans' massive reading process, which equip humans with the abilities to carry out future tasks in language understanding. With this intuition, we hypotheses that pre-training makes the distribution of trained language models approximate the humans'. The answer to this hypothesis is important because no matter how hard we try to create a "perfect" dataset for NLI, we could not avoid the biases alonewith the dataset. We need a more practical way to evaluate the effectivenss of our algorithms.

In order to test this hypothesis, we first collect the distribution of human's and different models' judgements of questions in SWAG. And we test the hypothesis with students' t-test. We propose the *human mean factor* by calculating the t-statistics between the model and humans. Then, we modify the models to Bayesian neural networks[13] for quantifying their uncertainties within different questions and endings. This allows us to carry out the hypothesis test in a variance level, and get the *human variance factor*.We also expect to carry out extensive experiments among different objectives of pre-training such as language modeling[9], machine translation[14], and discourse coherence[15] to compare their effects in language understanding in a same experiment setting, alone with other experiments to serve as a guide for using pre-training in the future language understanding tasks.

2 Methods

We propose a way to quantify the gap between trained and humans' language models, and experiment how pre-training bridges the gap. Our motivation is that, in many cases, we ultimately want a model that could understand language as well as humans did. This means, despite we design various datasets carefully, the datasets serve as the material for us to train models to be more aligned with humans' interpretation. Simply pursuing a high accuracy score without considering that is misleading. See Table 1, there exists a difference between humans' and model's judgments even if the algorithm picks the right answer.

2.1 Human Distribution Analysis

We first expect to collect humans' understanding for the questions in SWAG. We randomly sample 100 questions from SWAG and ask 10 students who are proficient in English at Nanjing University fill in the questionnaire below: After collecting the data, we get a discrete distribution for each question by averaging students' scores of each ending in the question. We will also get machines' score by predicting the best options.

Imagine that you are watching a video clip. The clip has a caption, but it is missing the final phrase. Please choose the best option, while at the same time rank the options from 1 to 10 degrees corresponding to how likely they are going to happen in a real world setting.
Example: Someone is shown sitting on a fence and talking to the camera while pointing out horses. Someone

- stands in front of a podium. (7)
- rides a horse using a hose. (3)
- is shown riding a horse. (9)
- ,the horse in a plaza field. (1)

Now, we calculate the Jensen–Shannon divergence, a smoothed version of Kullback–Leibler divergence between two distributions.

$$j_i = Jsd(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \tag{1}$$

$$D(P||M) = -\sum_i P(i)log(\frac{M(i)}{P(i)}) \tag{2}$$

$$M = \frac{1}{2}(P+Q) \tag{3}$$

From equation (1),(2),(3), we could get a vector of Jensen-Shannon distance with length 100, namely $J = \{j_i\}_{i=1}^{100}, 0 \leq j_i \leq 1$. We then rearrange the vector J to a distribution by count the number of each value falls into the regions $[0.1n, 0.1 + 0.1n], 0 \leq n \leq 9$. We now get a distribution of the distance between humans' language model and the chosen model, we denote its random variable as D_{model}. Here, we specify the D_{human} as a trivial random variable, i.e. $D_{human} = 0$.

We then perform hypothesis test among various models such as LSTM+GloVe, LSTM+ELMo, and BERT. We have reason to assume that the D_{model} approximately follows normal distribution since the probability that model predict exactly the same with humans or totally different with humans tend to be low, and most of predictions tends to cluster around a value. i.e. $\mathbb{E}(D_{model})$. We approximate this value with $\bar{x}_{model}$, which is the average of J. Equivalently, we approximate the variance of D_{model} with σ^2_{model} which is the mean square of the values in J. We choose the t-test in this case due to its advantage in testing small dataset.

Here we use LSTM+GloVe and LSTM+ELMo as an example. We denote their corresponding E and V as $\bar{x}_g, \bar{x}_e$, and σ_g, σ_e. Then, we get the t-statistics $t_{e,g}$ by the following formula:

$$t_{e,g} = \frac{10 * (\bar{x}_g - \bar{x}_e)}{\sqrt{\sigma_g^2 + \sigma_e^2}} \tag{4}$$

By choosing a certain p-value, we evaluate the null hypothesis H_0: ELMo does not make trained lanugage models further approximate the humans' compared to GloVe, in this example, respectively. We also propose the t-statistic between D_{human} and D_{model} as a metric to quantify the their distance.

$$t = \frac{10 * (\bar{x}_m)}{\sqrt{\sigma_g^2}} \tag{5}$$

We call it *human mean factor*.

2.2 Quantify uncertainties

When we say "I am no sure about this." or "I do not think it is possible.", we are expressing doubts. However, we actually can not assign this doubt a certain value, even if we try to do so. We can only evaluate it in a certain range. When it comes to the algorithm, like Table 1, it too confidently assigns a exact probability to each possible ending. With a higher probability, we understand that the model is more certain about the ending. However, we argue that there is another kind of uncertainties behind the exact the point estimate.[13]. For example:

An old man rides a small bumper car. Several people

- get in the parking lot. (76.58%)
- wait in the car. (15.28%)
- **get stuck with other bumper cars.** (6.75%)
- are running down the road. (1.39%)

Table 2: Example question answered by ELMo model, model assigned probability is in the parentheses.The right answer is bolded.

Answering the question requires the knowledge of "bumper car", which is a tiny car often appears in the fairground. Since this knowledge is difficult to extract from existing corpora [3], we assume that there should be a significant data uncertainty lying behind the exact probability.

This kind of uncertainty may vitiate the power of the *human mean factor* and we should include it in our consideration. We expect to see a good model not only is numerically high on *human mean factor*, but also have a strong grade on *human variance factor*. While we can get the uncertainties of humans by approximating it with the mean square error of the 10 testers, the crucial step is how could we quantify the uncertainties of algorithms.

We introduce Bayesian neural network to not only quantify the uncertainties but also hope to further improve state of the art.[16] Uncertainty can be divided into model uncertainty and data uncertainty according to the law of total variance. Denote x as the input, y as the output.

$$\begin{aligned} Var(y) &= Var(\mathbb{E}[y|x]) + \mathbb{E}[Var(y|x)] \\ &= model\ uncertainty + data\ uncertainty \\ &= U_m(y|x) + U_d(y|x) \end{aligned} \tag{6}$$

Model uncertainty Bayesian neural networks put a prior on the parameters $\mathbf{W}$ and aim to find the $P(\mathbf{W}|D)$, $D = (\mathbf{x}_i, y_i)_{i=1}^N$ is the given dataset. We also assume the function $f^{\mathbf{W}}(\mathbf{x})$ is used to predict the y value. Then we can specify the data is generated by:

$$y|\mathbf{W} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}), \sigma^2) \tag{7}$$

Notice that we need to find an approximation to the $P(\mathbf{W}|D)$ here since it is unfeasible to do the exact inference with conditioning on D. We use dropout variational inference method[17] here to avoid unnecessary changes to the original model. Replace $P(\mathbf{W}|D)$ with $Q(W)$, we get

$$p(y^*|x^*) = \int_{\mathbf{W}} p(y^*|f^{\mathbf{W}}(x^*))Q(\mathbf{W})d\mathbf{W} \tag{8}$$

At test time, we use Monte Carlo integration and get:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M}\sum_{j=1}^{M} f^{\widehat{\mathbf{W}_j}}(x^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 \tag{9}$$

Data uncertainty Here, we notice that we are in a classification setting, so we define $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ as functions that maps input x to a logit vector. We then transform the logit vector to a probability with softmax.

$$\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), diag(\boldsymbol{\sigma}(\mathbf{x})^2)) \tag{10}$$

$$\mathbf{p} = softmax(\mathbf{u}) \tag{11}$$

During the training phase, we optimized the negative log-likelihood for the dataset.

$$\mathcal{L}_{clf}(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} log \sum_{k=1}^{K} exp(u_{i,y_i}^{(k)} - log \sum_{c} exp\ u_{i,c}^{(k)}) - log\ K \tag{12}$$

The process of optimzing Equation(12) gets us the function $\boldsymbol{\sigma}$ and we could estimate the data uncertainty given input $\mathbf{x}^*$ with,

$$U_{d,i}(y^*|\mathbf{x}^*) = \sigma_i(\mathbf{x}^*)^2 \tag{13}$$

Here, the i represents the i-th ending in a question.

After quantifying the uncertainties of models, similar analysis should be carried out as Section 2.1, and we can get the *human variance factor* as another metric to evaluate models' proximity to humans' from the perspective of uncertainties.

Verifying the intuition We still remaining a intuition to verify as we raise in the very beginning that the process of pre-training is similar to humans' massive reading process, which equip humans with the knowledge to carry out future tasks in language understanding. We can study this intuition by summing up the data uncertainties that we calculate previously. And we should expect to see models with deep contextualized pre-training have less data uncertainties than other models.

2.3 Further analysis

Different objectives of pre-training What types of objectives of pre-training are most effective still remain unclear [12]. We hope to evaluate them such as language modeling, machines translation, and discourse coherence with our newly proposed two metrics.

The amount of fine-tuning data we need We expect to carry out experiments on BERT and GPT to test how many data the models need in order to get a satisfactory performance in the fine-tuning step.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*, 10 2018.

[2] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.

[3] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104. Association for Computational Linguistics, 2018.

[4] Dong Yu, li Deng, and George E Dahl. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. 12 2018.

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[6] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics, 2014.

[7] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[8] Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535. Association for Computational Linguistics, 2018.

[9] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[10] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics, 2018.

[11] Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. On the evaluation of semantic phenomena in neural machine translation using natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 513–523. Association for Computational Linguistics, 2018.

[12] Alec Radford. Improving language understanding by generative pre-training. In *arXiv preprint arXiv:1810.04805*, 2018.

[13] Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. *CoRR*, abs/1811.07253, 2018.

[14] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc., 2017.

[15] Yacine Jernite, Samuel R. Bowman, and David Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. 04 2017.

[16] Koray Kavukcuoglu Daan Wierstra Charles Blundell, Julien Cornebise. Weight uncertainty in neural networks. In *arXiv preprint arXiv:1505.05424*, 2015.

[17] Zoubin Ghahramani Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. In *arXiv preprint arXiv:1512.05287*, 2015.