LSTM Transformers and Word-Sense Disambiguation

Anthony Kimball, Katriana Davenport-Kimball

***Abstract*—Extensive knowledge of the world appears to be encoded, albeit perhaps vaguely, in the complex conditional structure of language models, once they have been trained on a sufficiently large sample of natural language. This research is an attempt to surface some of that information in a way which enables semantic inference to interpret text in context. The method chosen to this end consists of training a word-sense classifier using the hidden outputs of a sequence transformer neural language model trained over a large corpus at substantial expense, and made public by Google Research.**

I. Introduction

WSD (word-sense disambiguation) has been a problem of some concern in natural language processing effectively since before NLP was itself codified as a field. Some of the first descriptions of it in a computational context date back to the 1940s, and in a 1964 essay Bar-Hillel once argued that it could not be undertaken by a computer due to a perceived need for a complete world model before meaningful performance would be achieved. When statistical techniques began to filter into computational linguistics in the late 20th century, WSD benefited greatly from the shift in approach, and multiple generations of systems have grown out of the application of machine learning techniques.

A. Related Work

supWSD serves as a recent state of the art example [1]. Published by a group out of Sapienza University in Rome in 2017, its aim is to provide an API wrapper around a combined supervised WSD system and NLP pipeline. In addition to outputting competitive numbers on many common test data sets, it offers exceptionally fast training/testing times and high modularity, allowing for customization and additional layers as needed. However, by its nature as a supervised learning system, it requires large quantities of manually tagged data, which is effort-intensive to produce.

Prior to the publishing of supWSD, a group at Google theorized in 2016 regarding the application of neural network language models to the word-sense disambiguation problem with a semi-supervised approach [2]. They proposed a two-part system: one side a LSTM (long short term memory) model and the other a semi-supervised algorithm based on label propagation and sentence similarity. In the years since, several high-powered language models have been developed in the field, such as OpenAI's GPT-2 [3]. We followed the example of the 2016 paper for this investigation and chose to focus on the LSTM model BERT.

II. Methodology and System Structure

A. Introducing BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language model published by a group of Google researchers in 2018 [4]. Rather than relying on linear left-to-right or right-to-left training, it achieves bidirectionality (or, as they would put it, nondirectionality) through the application of attention model transformers [5] and a technique called MLM (Masked LM) to improve their training strategy. Out of the box, it pre-trains both a token-based strategy with MLM and a sentence-based strategy which is by default centered around a next sentence prediction task.

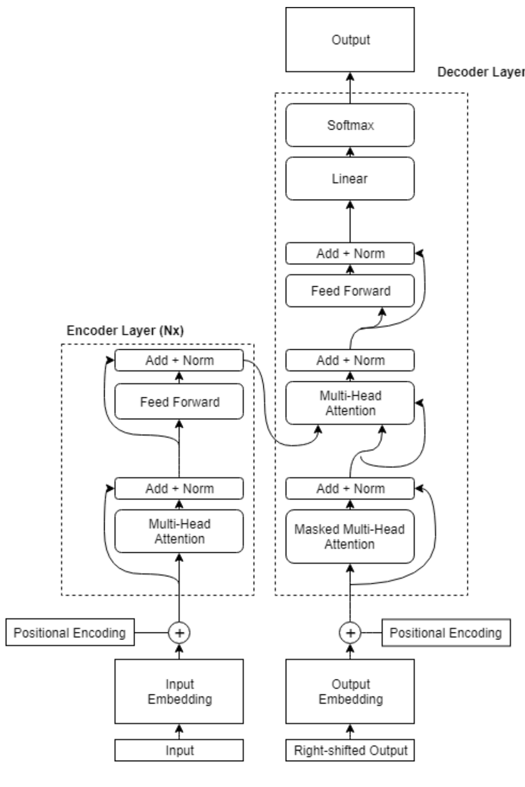

Fig. 1. The structure of an attention model transformer's encoder/decoder layer. (Based on a diagram from The Annotated Transformer [6])

In addition, BERT is designed in such a way as to facilitate the development and attachment of various 'head' modules fine-tuned to optimize performance on specific NLP tasks, several examples of which are given in the 2018 paper (such as specializations for named entity recognition and question answering). Our focus was therefore on developing a custom WSD head based on these examples and adjusting the pre-training regimen to best suit our needs for the task. The base code used here was drawn from the huggingface GitHub repository, implemented using PyTorch [7].

B. Training Alterations

By default BERT's MLM token-centric pre-training strategy involves masking and predicting 15% of all tokens in the input. After the selection of said tokens is done, 80% of them are replaced with a mask token, 10% with a random word, and 10% retain the original word. This set of ratios was chosen to, for example, prevent the model from adapting and optimizing itself purely for the task of predicting the masked token, and to prevent it from merely copying non-contextual embeddings for the 'correct' words.

We chose to use the pre-trained $\text{BERT}_{\text{SMALL}}$ language model, with additional training as best suited our purposes.

BERT has several examples of token-centric task heads, both in the original paper and in the PyTorch implementation used as a basis for this project. One of the primary examples given in the Google Research paper is named entity recognition, a classification task with some structural similarities.

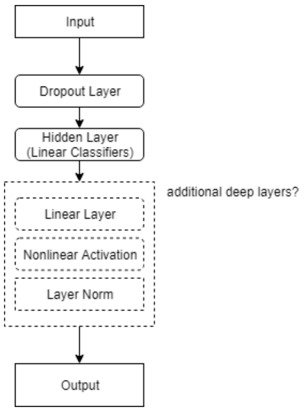

Fig. 2. WSD Structure

C. Classifier Features

The WSD head implemented for this project offers the option to extend the hidden BERT model outputs with WordNet categories as additional features. These are represented as a one-hot coding of 6 affordance categories corresponding roughly to parts of speech, which categorize the WordNet sense inventory used to tag our corpora. Another option implemented in the WSD head is an additional dense layer with non-linearity and layer normalization, preceeding the final dropout and classifier layers.

D. Training Performace

As presently parameterized, training one configuration of the WSD head (plain, wide, deep, or deep and wide) takes 45 minutes on a 4GB GTX1060 GPU.

III. Results

A. The WSDEval Test Set

Over the years, many test datasets have emerged for WSD systems. WSDEval combines all of these with WordNet sense annotations to create a unified evaluation schema for WSD systems [8]. It consists of five all-words datasets taken from the SemEval and Senseval task series, in addition to large SemCor and OMSTI training corpora for supervised systems, all of which have been modified to ensure a consistent data standard and tagged with the WordNet 3.0 sense inventory. We additionally acquired Google Research's available WSD corpora, consisting of SemCor and Masc documents annotated with NOAD word senses.

After standardizing the total corpus into a combination of XML sentences and WordNet sense keys, we had around 975MB of usable text data. From this total, we selected 0.3% of the sentences as test data and 0.8% as development data. Several cross-sections of the remainder were set aside for potential future training use, including medium (20%) and large (80%) training options (although our results do not currently rely on this due to time constraints: see below).

For our purposes the primary point of comparison is to supWSD, as the current state of the art. We acquired the most recent available version from the authors' GitHub repository and focused on making a comparison between our own system and supWSD running locally, on comparable hardware, with identical training and test datasets, the f-score results of which can be seen in Table I. Both systems were trained on an identical filtered version of the SemCor corpus. These numbers would, at the very least, seem to suggest that

System	Dev Data	Test Data	Unified Data
local supWSD	50.7	51.9	65.3
BERTForWSD (simple)	00.0	00.0	00.0
BERTForWSD (wide)	00.0	00.0	00.0
BERTForWSD (deep)	00.0	00.0	00.0
BERTForWSD (wide+deep)	00.0	00.0	00.0

TABLE I
TEST RESULTS: LOCAL SUPWSD VS. OUR DEVELOPED BERT HEAD ACROSS COLLECTED TEST DATA.

supWSD's performance may be overtuned on the WSDEval set, and its performance in a more generalized context should be studied more, or could be improved with adjustments.

B. Analysis

The test code evaluates the batch variance of F1 macro and micro scores, as well as global scores. One defect of the test code is that the logit threshold for the classifier boundary is determined using the test data. This is a bug. That boundary should be determined in training. In fact, it is currently a simple level boundary, but clearly much higher classification performance can be derived by training it conditionally, with another network, or some other mechanism. No investigation of the appropriate features for this task has yet been done.

IV. Conclusions

A. Towards Future Work

Although our discussion here focuses on the application of BERT, applying similar techniques with other language models may yield improved, or at least distinct, results. The obvious example is GPT-2, due to its recent fame/infamy if nothing else. In addition, the amount of time spent on regularizing the corpus data we used for these explorations means less time spent training. Further specialization of the pre-training and training schemes, or more iterations for them, could also be a viable direction for further investigations (and of course, more training data is also likely to improve performance).

It is of interest to learn the effect of WSD head depth on training time, responsiveness to added training data, and overfit threshold - which currently appears quite remote. Use of a reflective attention mechanism or adversarial network to implement active learning, in order to focus the traning process on the classification margin may improve convergence.

Additional layers added to the BERT head interface may also improve the results. Our current code allows for deeper layers with a nonlinear component, as shown in Figure 2, but time was not available for a fuller examination and comparison of what such deeper layer stacking could achieve. Even without additional layers, increasing the number of parameters used in the model may prove helpful, as this would reduce the likelihood of becoming 'caught' in a local minimum during optimization due to a higher probability of local minima in subspace becoming saddle points in their enclosing feature space.

Finally, the WSD task may be illustrative of complex conditional structure being latent in the pre-trained language model, but it does little to elucidate the content or structure of that knowledge. It would be of great interest to explore ways in which that implicit knowledge can be make explicit, or otherwise available for reasoning.

V. Bibliography

References

[1] S. Papandrea, A. Raganato, and C. Delli Bovi, "Supwsd: A flexible toolkit for supervised word sense disambiguation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Copenhagen, Denmark, 2017.

[2] D. Yuan, R. Doherty, J. Richardson, C. Evans, and E. Altendorf, "Word sense disambiguation with neural language models," *CoRR*, vol. abs/1603.07012, 2016. [Online]. Available: http://arxiv.org/abs/1603.07012

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[6] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017. [Online]. Available: https://doi.org/10.18653/v1/P17-4012

[7] "Pytorch pretrained bert: The big and extending repository of pretrained transformers," 2019. [Online]. Available: http://github.com/huggingface/pytorch-pretrained-BERT

[8] A. Raganato, J. Camacho-Collados, and R. Navigli, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *Proceedings of EACL*, Valencia, Spain, 2017.