TTCB System Description to a Shared Task on Implicit and Underspecified Language 2021

Peratham Wiriyathammabhum
peratham.bkk@gmail.com

Abstract

In this report, we describe our Transformers for text classification baseline (TTCB) submissions to a shared task on implicit and underspecified language 2021. We cast the task of predicting revision requirements in collaboratively edited instructions as text classification. We considered Transformer-based models which are the current state-of-the-art methods for text classification. We explored different training schemes, loss functions, and data augmentations. Our best result of *68.45%* test accuracy (*68.84%* validation accuracy), however, consists of an XLNet model with a linear annealing scheduler and a cross-entropy loss. We do not observe any significant gain on any validation metric based on our various design choices except the MiniLM which has a higher validation F1 score and is faster to train by a half but also a lower validation accuracy score.

1 Introduction

A shared task on implicit and underspecified language 2021 is the first installment of predicting revision requirements in collaboratively edited instructions (Bhat et al., 2020) based on the wikiHowToImprove dataset (Anthonio et al., 2020). The dataset consists of sentences and their revisions if any. There are *5* rule-based revision types which are pronoun replacement, 'do' verb replacement, verbal phrase compliment insertion, adverbial and adjectival modifier insertion, and logical quantifier or modal verb insertion. The task is to determine whether a given sentence with its corresponding context paragraph needs any revision based on the aforementioned revision types.

Previous work (Bhat et al., 2020) compares BERT (Devlin et al., 2019) and BiLSTM on the full wikiHowToImprove dataset which has *2.7* millions sentences. The previous experiment integrates *4.25* millions of unrevised sentences from wikiHow to further balance the training set. Their results suggest BERT over BiLSTM. Our systems build upon this finding and further explore Transformer-based models.

Table 1: Example instances from the wikiHowToImprove dataset. The first sentence does not require any revision. The second sentence needs a revision by replacing the pronoun 'They' with the word 'Meeting' to provide more clarity.

Sentence	Label
Do not pour the petals in the perfume on storing .	KEEP_UNREVIS
They also give managers the opportunity to tell everyone the same thing at once , which can cut down on gossip .	REQ_REVISION

The codes for our systems are open-sourced and available at our GitHub repository[1].

2 Models

2.1 XLNet

XLNet (Yang et al., 2019) is the current state-of-the-art for text classification on various benchmarks such as DBpedia, AG, Amazon-2, and Amazon-5. XLNet is an autoregressive Transformer language model which further explores longer context modeling to capture long-term dependencies between words. We consider the HuggingFace Transformer library (Wolf et al., 2020) for our experiments on XLNet.

2.2 Siamese training

Siamese model training (Bromley et al., 1993) is an off-the-shelf neural-networks training paradigm that learns similarity embedding for verification by using two identical neural networks to extract

[1] https://github.com/perathambkk/unimplicit_shared_task_acl_2021

Proceedings of the 1st Workshop on Understanding Implicit and Underspecified Language (UnImplicit 2021), pages 64–70
Bangkok, Thailand (online), August 5, 2021.

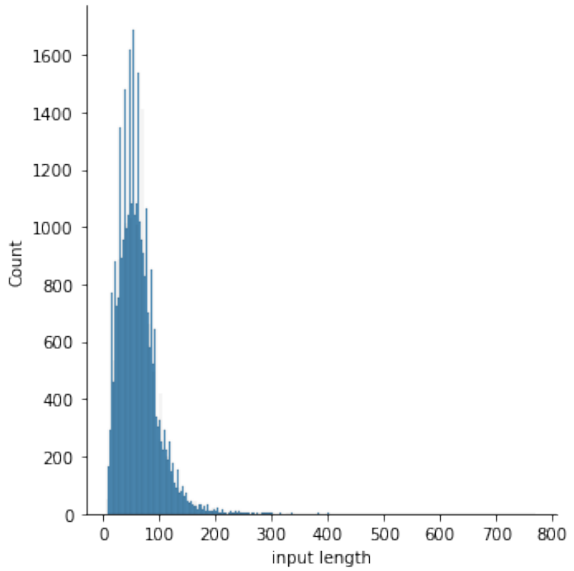

Figure 1: **The distribution of the input length derived from the shared task training set.**

feature vectors for a threshold-based input pair comparison. The model is learned from the signal whether an input pair is similar or dissimilar. This approach has been shown in various settings to produce a good vector embedding space. We consider the sentence-Transformers library (Reimers and Gurevych, 2019) for our experiments on Siamese training.

3 Experimental Setup

Our input is a simple concatenation of a sentence and its context paragraph. We tried different context lengths and found that *128* yields the best result. From Figure 1, the mean input length is only *62.58* with the standard deviation of *36.00*. This is from the shared task dataset which is the subset of the original wikiHowToImprove dataset and has *45,909* sentences in total (*39,187* sentences in the training set.). The statistics suggest setting the context length less than *200* to be cost-effective and there are only *1,632* training instances (around *4%*) having their input lengths longer than *128* with the maximum length of *770*.

All of our experiments were done in the Google Colab setting. We used only base models for all Transformers. We used the batch size of *8* and the learning rate of *1e-5* for all experiments. We considered linear annealing scheduler since other schedulers, such as ReduceLR scheduler, cosine annealing scheduler, or cosine annealing scheduler with restart, do not provide any significantly different results. Also, adding a warm-up step does not make any difference too. We trained the model for

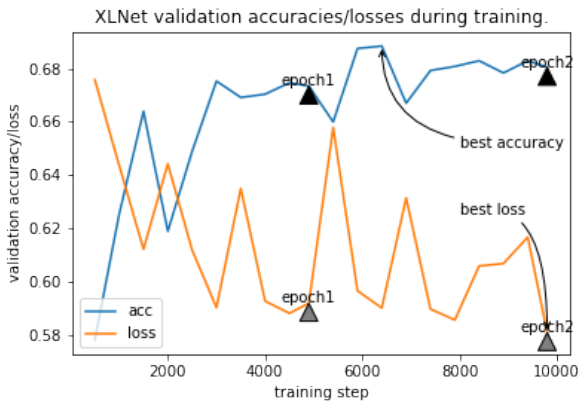

Figure 2: **Validation accuracies and losses during training of the XLNet model.**

Table 2: Development accuracies of text classification Transformer models. Majority means always predicting using the majority class label which is either always positive or negative in this balanced development set.

Model	Dev Accuracy
Majority	50.00
OpenGPT-2	65.50
XLNet	**68.84**
Bigbird	68.69

4 epochs (following the standard fine-tuning procedure in the original BERT paper (Devlin et al., 2019) which recommends *2-4* epochs.) and sample a model state at every *500* training steps for evaluation on the development set. Most of the best models are from the second epoch. This step helps to save the best model parameter state which could be empirically up to *1%* better in development accuracy than only collecting the model state at the end of each training epoch as depicted in Figure 2 for XLNet.

3.1 Text Classification

We compare XLNet with OpenGPT-2 (Radford et al., 2019) and Bigbird (Zaheer et al., 2020) for text sequence classification in Table 2. OpenGPT-2 is an unsupervised multitask language model. Bigbird is a recent state-of-the-art text classification model on some benchmarks, such as arXiv (He et al., 2019), Patents (Lee and Hsiang, 2020), or Hyperpartisan (Kiesel et al., 2019). Bigbird utilizes better computation methods to efficiently model longer sequence lengths than XLNet. The results suggest that modeling longer sequence length than a sentence helps as seen in XLNet and Bigbird, however, Bigbird is only comparable to XLNet in terms of accuracy.

Table 3: Development accuracies of different loss functions on the XLNet model.

Loss Function	Dev Accuracy
binary cross-entropy (BCE)	**68.84**
label smoothed BCE	68.78
cost-sensitive BCE	68.81
cost-sensitive multiclass CE	67.80

Table 4: Development accuracies of data augmented Bigbird.

Augmentation	Dev Accuracy
Bigbird	**68.69**
+ negative class augmentation	64.74
+ cost-sensitive BCE	68.47

3.2 Loss Functions

Label smoothing (Szegedy et al., 2016) is a design choice in loss function which helps improve the model performance in many tasks by smoothing the cross-entropy label loss from *0/1* to α/K for other classes and $(1-\alpha)$ for the target class using an arbitrary hyperparameter α. We used the α value of *0.1*.

Previous work (Bhat et al., 2020) also emphasizes the class-imbalance issue in this task. Therefore, we tried cost-sensitive cross-entropy loss to weigh more on the positive class (revision needed) which suppose to have more information. We weighted the positive class by *0.6* and the negative class by *0.4*. We also tried cost-sensitive multiclass cross-entropy loss where we train on revision types as the label set and convert them to *0/1* for prediction with the hope that the model might better learn the structure in the data. We weighted each class by the inverse of its number of instances.

The results in Table 3 suggest that there might not be any significant class-imbalance issue that can be alleviated via various cost function design choices since the development accuracies are very much the same. The exception is the multiclass setting where we conjecture that that revision types might make the training task harder instead.

3.3 Data Augmentation

The shared task data provide the revisions when the labels are positive (revision needed) so we tried to generate more data from these. We assumed the revised sentences provide more signals of no revision required. Therefore, we simply put the negative label on those sentences. We hoped that these data instances will provide more useful learning signals when added to the training set as more informative negative instances. Our reason is it should be more certain that most revised sentences should not require revisions, at least from the revised type. From Table 4, we chose Bigbird since it is more computationally efficient. However, adding more data does not improve the performance. Instead, the performance decreases to *64.74%* accuracy. Still, adding cost-sensitive binary cross-entropy can bring the accuracy back to be comparable to a vanilla Bigbird. This indicates that cost-sensitive loss may be helpful if we were to perform data augmentation. The cost-sensitive binary cross-entropy loss function adds a scalar weighting w to the cross-entropy loss term for each class.

$$loss(x, class) = w[class] \cdot (-x[class] + \log\left(\sum_j (exp(x[j]))\right)) \quad (1)$$

Since the revision types are based on syntax, we also tried to add more syntactic information to the models. Our preliminary attempt is to add part-of-speech tags and dependency trees (tagged using spaCy (Honnibal et al., 2020)) as additional context inputs by concatenation to existing sentence and context inputs. However, they do not provide any useful learning signals as also observed from recent attempts to learn syntactic Transformers. We also tried to learn solely from part-of-speech tags and dependency trees inputs and they provide very low accuracies similar to random. Many recent studies (Clark et al., 2019; Hewitt and Manning, 2019; Rogers et al., 2020) also show that BERT learns some syntactic information during its pretraining steps. However, there are still some works (Sundararaman et al., 2019; Wang et al., 2020a) showing that explicitly adding syntactic information may still improve BERT or Transformer performance.

Table 5: Development accuracies and F1 scores on CrossEncoder or BinaryEncoder for text classification.

Model	Dev Accuracy	F1 Score
XLNet	**68.84**	70.08
MiniLM-L-12	68.44	71.72
Siamese-BERT	63.57	69.77

3.4 Siamese Training

To begin with, the sentence-Transformers library (Reimers and Gurevych, 2019) supports both CrossEncoder (the same architecture for text classification) and BiEncoder (Siamese training). We tried their CrossEncoder model with MiniLM-L-12 model (Wang et al., 2020b) pretrained on ms-marco (Nguyen et al., 2016) for passage reranking

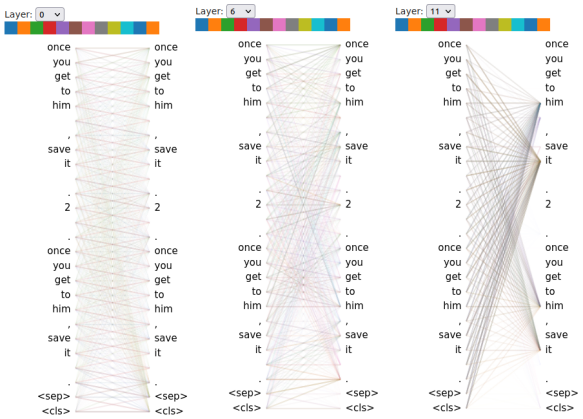

Figure 3: **BertViz XLNet attention-head visualization from the first attention head of layers {*1, 7, 12*} for a revision-required sentence, 'Once you get to him, save it.'**

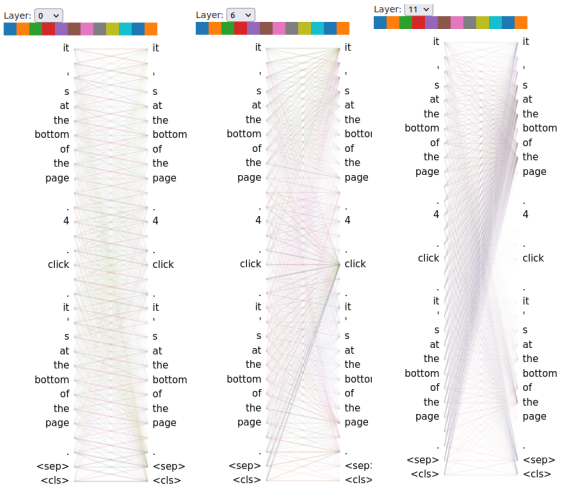

Figure 4: **BertViz XLNet attention-head visualization from the first attention head of layers {*1, 7, 12*} for a no-revision-required sentence, 'It's at the bottom of the page.'**

(slightly after the competition). The results in Table 5 indicate a lower development accuracy for MiniLM-L-12 but a comparable F1 score. The advantage of MiniLM-L-12 is its training cost is less than half of the XLNet model. We observed the speed-up on an NVIDIA-K80, an NVIDIA-P100, and an NVIDIA-T4 GPU from Google's Colab in our experiments. MiniLM is more lightweight and may be suitable for faster research cycles in general. Next, we depict our results on vanilla Siamese-BERT. We speculate that sentence embedding models have effortlessly good F1 scores because of their higher recall based on the nature of embedding vector spaces.

3.5 Visualizing XLNet

We consider BertViz (Vig, 2019) to explain the XLNet model via attention visualization. Figure 3 shows the attention weights from layers {*1, 7, 12*} for a revision-required input sentence from the development set, 'Once you get to him, save it.' The visualization suggests that early layers learn simple and local patterns while middle layers learn longer dependencies and the top layers learn revision patterns. This is from the rightmost plot which shows large weights on the terms, 'him' and 'it', which probably require revisions.

Figure 4 shows another example from a no-revision-required input sentence from the development set, 'It's at the bottom of the page.' The early and middle layers exhibit similar patterns as the previous example which are local or longer dependencies. However, the top layers show even weighting for each word in the input sentence which instead does not indicate any revision signal. From the model views which show all attention heads in all layers in Figure 5 and Figure 6, the visualizations suggest that different attention heads from the same layer exhibit similar patterns.

4 Conclusion

This report describes our baseline systems for a shared task on implicit and underspecified language 2021, predicting revision requirements in wikiHow. Our best result is from the XLNet model with a linear annealing scheduler and a cross-entropy loss. We do not observe any significant gain on any validation metric based on our various design choices. The cost-sensitive loss might help only when performing data augmentation. MiniLM is comparable to XLNet but at a half computation cost. We summarize the results as finetuning Transformer-based language models for text classification only provides incremental improvements even though better language models consistently lead to better results. Also, the accuracies at most $\sim 70\%$ are not very practical. This suggests a big challenge for the language models in the context of implicit and underspecified language. We release our training code as an unofficial baseline for the challenge.

There are many possible future directions. First, we have not considered any advanced loss functions, such as Triplet loss (Weinberger et al., 2005; Hoffer and Ailon, 2015), for our Siamese training experiments. Second, recent work on predicting revisions in wikiHow (Debnath and Roth, 2021) depicts a promising integration of syntactic preprocessing and sentence embedding training. Nevertheless, more data analysis is needed to pinpoint what a particular model should learn.

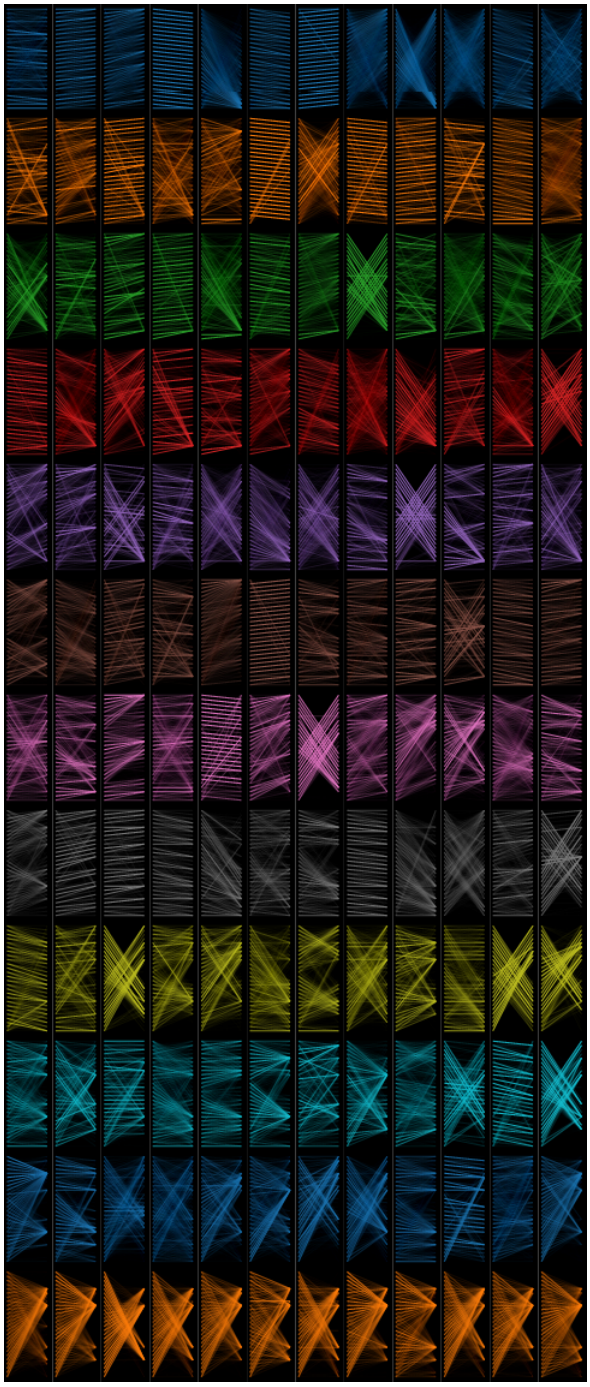

Figure 5: **BertViz XLNet model-view shows all attention heads from all layers for a revision-required sentence, 'Once you get to him, save it.' Each row corresponds to a layer and each column corresponds to an attention head.**

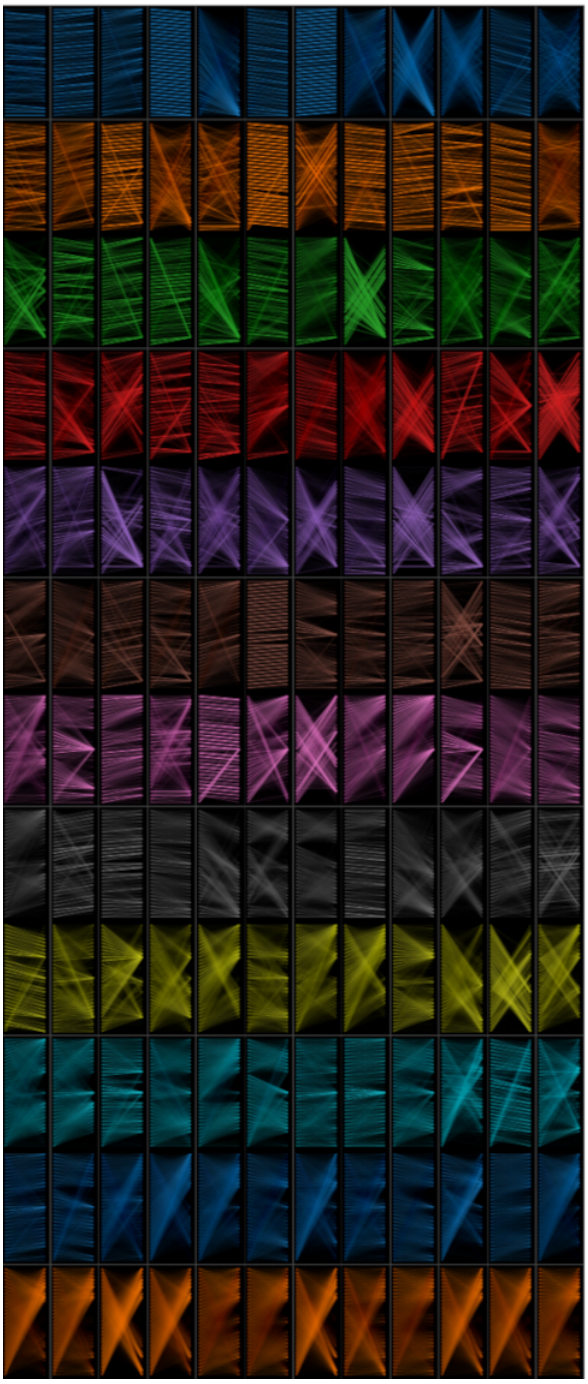

Figure 6: **BertViz XLNet model-view shows all attention heads from all layers for no-revision-required sentence, 'It's at the bottom of the page.' Each row corresponds to a layer and each column corresponds to an attention head.**

Acknowledgments

We would like to thank anonymous reviewers for their constructive feedback.

References

Talita Anthonio, Irshad Bhat, and Michael Roth. 2020. wikiHowToImprove: A resource and analyses on edits in instructional texts. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5721–5729, Marseille, France. European Language Resources Association.

Irshad Bhat, Talita Anthonio, and Michael Roth. 2020. Towards modeling revision requirements in wikiHow instructions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8407–8414, Online. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6:737–744.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Alok Debnath and Michael Roth. 2021. A computational analysis of vagueness in revisions of instructional texts. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 30–35.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. 2019. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839.

Jieh-Sheng Lee and Jieh Hsiang. 2020. Patent classification by fine-tuning bert language model. *World Patent Information*, 61:101965.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42.

Jixuan Wang, Kai Wei, Martin Radfar, Weiwei Zhang, and Clement Chung. 2020a. Encoding syntactic knowledge in transformer encoder for intent detection and slot filling. *arXiv preprint arXiv:2012.11689*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. 2005. Distance metric learning for large margin nearest neighbor classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pages 1473–1480.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33.