Improving Deep Learning Model Robustness via Adversarial Attack

Jack Scudder
Harvard University
jscudder@g.harvard.edu

Diego Zertuche
Harvard University
diego_zertucheserna
@g.harvard.edu

Vikram Shastry
Harvard University
vgs298@g.harvard.edu

Moses Mayer
Harvard University
mosesmayer
@college.harvard.edu

Abstract

Deep machine learning models have been shown to be vulnerable to adversarial attacks, which can fool state-of-the-art models by strategically modifying an input example. SOTA models have performed well on a variety of NLP tasks, yet, when tested against adversarial examples the top published models see a significant decrease in performance, indicating a lack of robustness in these models. Our methodology perturbs inputs to LSTM and BERT models using a variety of published attacks from `TextAttack`. For training we use a mixture of pre-processed perturbed and clean examples. We seek to improve model robustness by finding an optimal mix of examples for a given attack. Our results indicate, among other things, that (1) a BERT-based model benefits more from this approach when compared to an LSTM model and (2) TextFooler has the highest performance among the chosen methodologies.

1 Introduction

Szegedy et al. (2014) published their seminal work revealing inherent weaknesses in neural networks which ushered in an entirely new area of machine learning for computer vision (CV). Three years later Jia and Liang (2017) showed that NLP models were not immune. Previously published NLP models developed for reading comprehension and question answering tasks had achieved high accuracy and F1 scores on common benchmark datasets. Yet, when tested against adversarial examples by Jia and Liang and Wang and Bansal (2018), the top published models for the Stanford Question Answering Dataset (SQuAD) (2016) saw a significant ($\geq 50\%$) decrease in F1 scores, indicating a lack of robustness in these models. Although BERT (Devlin et al., 2019) and its variants have shown near-human performance for myriad tasks, these pre-trained language models are not sufficiently robust to adversarial attack either (Jin et al., 2020).

Researchers have tried many avenues to study and remedy these architecture weaknesses in neural nets. The different approaches distill into two primary categories: (1) attacks and (2) defenses. For attacks, researchers design adversarial attacks to determine where NLP models fail and how much the accuracy or F1 score degrades. Examples of named attack methods include **DeepWordBug** (Gao et al., 2018), **PWWS** (Ren et al., 2019), **TextFooler** (Jin et al., 2020), and **BAE** (Garg and Ramakrishnan, 2020) each of which perturbs input text with the goal of example misclassification.

For adversarial defenses, researchers seek to improve model robustness to different published attacks. These defenses can include methods ranging from creating upper bounds of worst-case loss (Jia et al., 2019) or adding a pre-processing step to modify inputs before passing them through the model (Wang et al., 2021).

Susceptibility of NLP models to attack is a problem worth investigating for several reasons. We increasingly rely on NLP models to handle important search queries, financial transactions, and home security features. In all roles - whether a researcher, a consumer, an engineer - we need to have trusted, predictable results from the models we use.

With these considerations in mind, we were initially drawn to the min-max saddle-point formulation described by Madry et al. (2019). It showed impressively robust results for CV, and we hoped to adopt the same approach for NLP. The challenge was applying a method well-suited for continuous pixel representations in CV to discrete symbols in NLP. Instead of a min-max optimization problem, we opted for a revised approach.

Adversarial attacks are well documented in NLP research literature, so we took a small cross-section of five attacks and used them to generate adversar-

ial examples to improve *robustness*. By mixing clean and adversarial examples, we aimed to find an optimal mix that would lead to a trained model more robust to adversarial attack. Our goal: find a percentage mix of perturbed and clean examples with different attacks to try to find the "best" combination of training examples for robust classification. We conducted experiments on the **Rotten Tomatoes (RT)** dataset (Pang and Lee, 2005) on a binary sentiment analysis classification task, which is common in adversarial NLP research.

Our contributions are as follows:

- We provide a short survey of popular adversarial attacks in NLP.
- We compare published attacks against a naive attack for analysis.
- We establish baseline results for further adversarial training research.
- We propose new directions for future work.

2 Related Work

Building upon the work of Jia and Liang (2017), Wang and Bansal (2018) propose a novel adversarial-generation algorithm that increases the variance of the adversarial training data, to make it more effective to train on adversarial examples and reduce the specificity of the original method. This work also introduces a method of incorporating semantic relation features to a model for increased robustness. We used this concept of adversarial training in general, while differing our method by comparing different attacks developed since 2018 and using BERT as our primary model of consideration.

After developing our baseline attack – hereafter referred to **Baseline Method (BLM)** – and gathering results, we found a paper by Sato et al. (2018) that introduced "interpretable adversarial training" or iAdvT, which very closely resembles our independently developed BLM attack. They use a random perturbation to decide a direction and then find the nearest true word as a perturbation. Our implementation slightly differs, but we wanted to include reference to it.

All pre-defined attacks were implemented with the Python library `TextAttack` (Morris et al., 2020), a framework to analyze and leverage results from adversarial attacks. It consists of modular PyTorch code that works with HuggingFace libraries and has "Attack Recipes" of implemented published attacks. Each attack is broken into four components: a transformation, constraints, a goal function, and a search method. All attacks we used adhere to the following constants:

- Constraints - no stopwords are modified.
- Goal function - untargeted classification: change positive reviews to be classified negatively and vice versa.
- Search Method - word importance ranking: rank words according to how they impact classification. Each attack has a slightly different approach; examples include deleting words (TextFooler, BAE) or using weighted word saliency (PWWS) to evaluate each word's impact on classification.

Our work compares different published attacks, with attack modes concisely explained below.

DeepWordBug (Gao et al., 2018) determines word importance then greedily swaps characters within those important words until the classification changes. The authors published it before BERT existed, but it works on BERT models.

Probability Weighted Word Saliency (PWWS) by Ren et al. (2019) calculates the degree of change in classification if a word is removed, then greedily replaces important words with their WordNet synonyms. It was one of the first synonym substitution attacks.

TextFooler (Jin et al., 2020) has become a baseline attack for BERT models. When changing words to fool a text classifier, TextFooler ensures swapped words meet a certain threshold for semantic similarity *and* it requires that the resulting adversarial sentence is also semantically similar to the original sentence.

Bert-based Adversarial Examples (BAE) (Garg and Ramakrishnan, 2020) used TextFooler as a benchmark and changed the method by applying BERT's [MASK] token to mask important words and subsequently using the masked language model functionality to predict new words. Garg and Ramakrishnan (2020) uses a similar approach to ours in that it mixes perturbed and clean examples to observe accuracy; however, notably their goal is different than ours. They seek to develop an attack, whereas our goal is model robustness from attack.

3 Methodology

Robustness via Adversarial Attack. Our method is captured below with the primary goal of finding (1) a percentage mix of perturbed and clean examples, (2) with different attacks to try to find the

"best" combination of training examples for robust classification. The steps for our method:

1. Select popular attacks implemented within `TextAttack`. Add our naive attack, BLM, for comparison.
2. Preprocess examples: create dataframes of perturbed examples for each attack.
3. Create dataloaders with perturb percentages of $0 - 100\%$ in 10% increments.
4. Train a unidirectional LSTM and BERT-base-uncased on different combinations of clean & perturbed examples.
5. Evaluate accuracy on held-out clean test data: find optimal mix of attack-with-perturbed data.

Attack Methods. The pre-defined attack methods are described in Section 2 with their citations. Additionally, our naive baseline method for perturbing input examples, BLM, is described here.

BLM uses a pre-created dictionary that has $k = 10$ nearest neighbors for every word in the `word2vec` (Mikolov et al., 2013) embedding space. When creating the perturbed example dataframes, 25% of the words were randomly swapped with one of the $k = 5$ nearest neighbors utilizing weighted probabilities based on the cosine similarity scores of each candidate with the original word. As referenced in Section 2, BLM is similar to Sato et al. (2018).

4 Experiments

Task and Dataset. For sentiment classification, we use the **Rotten Tomatoes (RT)** dataset by Pang and Lee (2005) which features movie reviews labeled as positive or negative. RT has 10k total examples with an 80%-10%-10% train-validation-test split. It is publicly available and 1.75 MB in size. We used the HuggingFace `Datasets` (Lhoest et al., 2021) and `Transformers` (Wolf et al., 2020) libraries to load and preprocess the data.

Originally we planned to use the IMDB dataset (Maas et al., 2011) as well, but processing RT took 50% of our AWS credits, and our least computationally intensive attack (DeepWordBug) took over 100 hours on the 25k IMDB training examples. We curtailed our scope to complete the project.

Models. We used two models in our experiments.

1. A unidirectional LSTM with a hidden layer size of 256. Throughout our work we refer to it as an LSTM model trained on a training set with $p\ \%$ of examples perturbed.
2. A BERT model, `bert-base-uncased`, from HuggingFace `Transformers`, which was pretrained on BookCorpus and the English Wikipedia datasets, described in the original BERT paper (Devlin et al., 2019).

Setup. Following the method from Section 3, we saved dataframes of perturbed and clean examples for model training. We structured our dataloaders so that if a perturbed example was chosen for training, *its corresponding example was not used*. This way we did not pass the "same" example through the model twice.

After creating the perturbed dataframes, each model was trained with a different split of the clean/perturbed datasets. For a 10% mix (as seen in Table 1), 10% of the examples are randomly selected from the perturbed example column. This new data mix stays the same throughout all epochs of model training. Based on best empirical validation accuracy, models trained for 4 epochs with a learning rate of $2 * 10^{-5}$, as suggested in the original BERT paper (Devlin et al., 2019). We used the ADAM optimizer (Kingma and Ba, 2014) and Binary Cross Entropy as our loss function.

5 Results and Discussion

Test set accuracy after training each model-attack-example combination is shown in Table 1 and Figure 1. Broadly speaking, training on perturbed examples helps in test performance, as 8 out of 10 of model-attack combinations have a peak at a non-zero percentage of perturbed examples.

The BERT model benefits more from perturbed examples during training, as the performance gain against the clean dataset is stronger than in the LSTM. The best percentage of perturbed examples for the LSTM is generally around 10%, whereas the BERT model performs best around 20% to 30%. These results confirm what we know about BERT: it is far more complex than an LSTM, relies more on context due to its transformer architecture, and generalizes better to out-of-sample examples.

TextFooler has the best overall performance in the BERT model at 10% of perturbed examples, which achieves a 86.5% accuracy score for the test set. The peak performance for the LSTM is achieved with the clean dataset, again confirming that the LSTM model is not complex, hence why we used it as a results baseline. Using adversarial attacks to increase model robustness is most

Model	Attack	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
BERT	BAE	84.3	82.3	**84.4**	80.3	81.5	81.5	82.2	77.7	79.0	75.6	74.0
	DWB	49.0	84.0	83.4	**84.1**	51.0	82.8	78.6	79.2	78.3	73.5	49.0
	BLM	83.8	82.8	84.4	84.7	83.3	84.9	83.8	**86.0**	82.0	84.1	83.6
	PWWS	51.0	83.3	**83.8**	82.5	82.0	82.7	82.2	82.2	77.7	71.0	66.9
	TF	86.2	**86.5**	84.4	82.7	81.6	81.7	81.4	80.4	73.3	49.0	53.8
LSTM	BAE	**67.2**	64.5	63.6	63.8	62.0	62.0	64.2	62.3	62.2	61.3	60.1
	DWB	65.3	**65.8**	64.8	60.2	60.2	61.7	57.0	59.5	58.9	57.1	55.4
	BLM	63.8	**67.0**	65.3	65.5	65.3	63.9	62.6	66.2	66.0	63.6	63.5
	PWWS	**68.0**	64.3	64.6	65.9	65.3	61.7	62.2	61.7	62.4	58.4	57.5
	TF	62.7	**64.8**	64.0	63.1	62.6	58.8	59.5	60.4	61.0	59.6	56.6

Table 1: Test set accuracy for each attack/model pair and perturbation percentage. "Perturbation percentage" represents the number of perturbed examples based on the given attack. Ex: the 20% column equates to 20% perturbed examples and 80% clean examples. Abbreviations for attacks: **BAE** = Bert-based Adversarial Examples; **DWB** = DeepWordBug; **BLM** = Baseline Method; **PWWS** = Probability Weighted Word Saliency; **TF** = TextFooler.

effective when using large, complex models.

Our BLM attack is as effective as the other attacks explored. Using perturbed examples created with this attack improves test performance in both the BERT and the LSTM models. This shows that to increase the robustness of a model, a complex adversarial attack is not entirely needed, and a naive approach such as swapping a word with one of the closest neighbors is enough to increase the robustness of a model. There are strange dips in performance in some of the BERT runs (see Figure 1). We hypothesize that this happened because of (1) the random initializations of the models and splits, and (2) the dataset size. RT is a very small dataset in NLP, so a poor split could truly hinder performance. To combat this in the future, we will run experiments multiple times to improve confidence in our results.

6 Conclusion and Future Work

Conclusion. We present a method that compares model performance on perturbed examples from published attacks. Our results confirm that training with adversarial examples can improve model robustness, most significantly in BERT models. The highest accuracy in our experiments came from a 10% TextFooler perturbation. The relative success of our BLM indicates that a full attack isn't necessary to improve robustness; a simple word swap will have the same effect.

Future Work. Our research revealed multiple potential areas for future work. The three most salient ideas follow. First, we can enhance the evaluation of our robust models. After a single iteration of training with perturbed examples from our current experiments, we could evaluate the trained models under attack. By stacking adversarial training with adversarial attacks, we would improve our analysis and metrics for robustness.

Second, using additional datasets would augment our training process. Specifically, larger datasets coalesced with additional learning tasks would lead to more convincing results. Although seemingly obvious, we wanted to note that computational cost and individual inexperience led us to reduce our project scope from its original state.

Third, for a new direction entirely, adversarial training is a "feedback loop". A more sophisticated model necessitates more sophisticated attacks, which in turn necessitates additional training, and so on. Determining how to control this feedback loop is a major unsolved question. From an operational standpoint, thinking like an adversary may be the best approach for developing a robust adversarial detection model. Such a detection model may be more useful than rote attempts at improving model robustness.

7 Impact Statement

Adversarial attacks are becoming an increasingly important aspect in natural language processing. Recent developments in both computational power and model sophistication have enhanced the power of adversarial learning. These developments are not ipso facto "good" or "bad." Rather, they increase the potential for misuse. The adversarial attacks

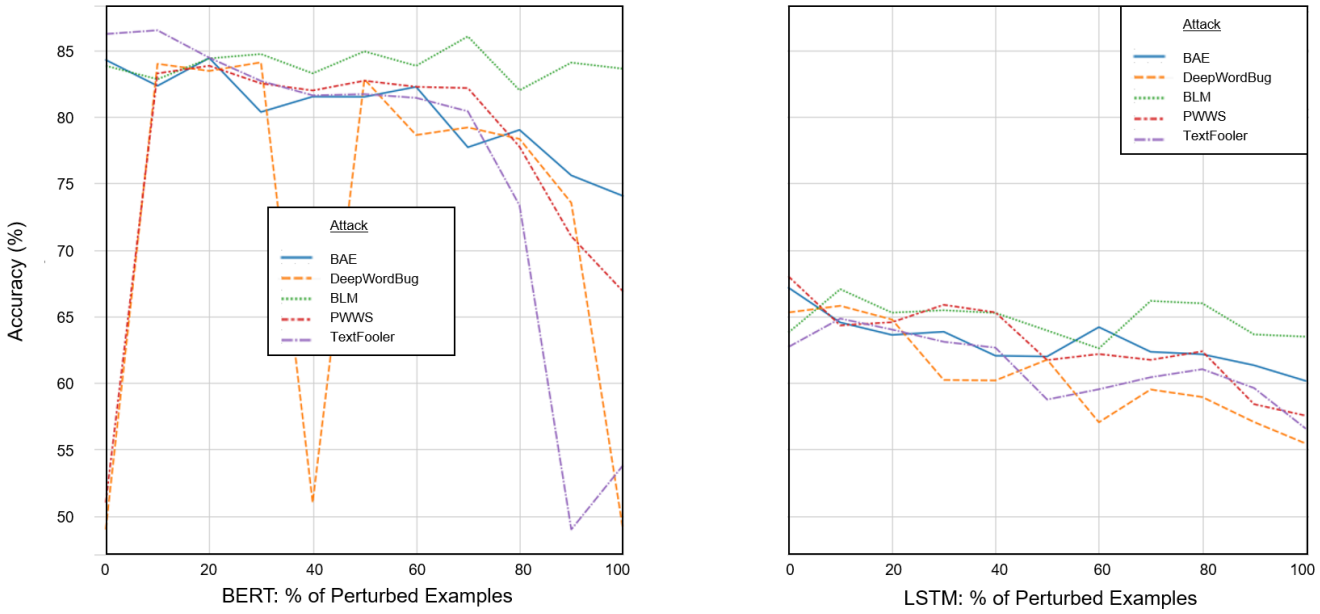

Figure 1: Plots of test set accuracy for each attack and perturbation percentage for each model. "Perturbation percentage" represents the number of perturbed examples based on the given attack. Ex: 20% perturbed examples implies 80% clean examples. Abbreviations for attacks: **BAE** = Bert-based Adversarial Examples; **BLM** = Baseline Method; **PWWS** = Probability Weighted Word Saliency

we used for our experiments have been shown to change the outcome of sentiment analysis on restaurant and film reviews. A business competitor could potentially use this technology to unfairly harm a rival. And this form of misuse can have broader ramifications.

In their work *The Coddling of the American Mind: How Good Intentions and Bad Ideas Are Setting Up a Generation for Failure*, Greg Lukianoff and Jonathan Haid note that "[in a polarization spiral, for every action, there is a disproportionate reaction. Prior to the polarization era, enthusiasm for one's group or team was a strong motivation. How, the motivation is hatred for the opposition. It is as if we are holding signs saying, 'Tell me horrible things about the other side, I'll believe anything!' Americans are now easily exploitable. And a large network of political entrepreneurs, for-profit media sites, and foreign intelligence agencies are taking advantage of this.]" One such manifestation of this is fake news detection, which has become a topic of significant interest in the aftermath allegations of foreign meddling in the 2016 US presidential election. Adversarial attacks can be used to counteract bot detection mechanisms and thereby allow implementation of information warfare from a nefarious actor.

Thus, if used as intended, the mechanisms described in this paper can have potentially far-ranging consequences. The danger in this technology lies in the intent of the user. The technology itself, as mentioned earlier, is neither "good" nor "bad." Rather, its deployment is what can have ramifications when the technology is used as intended. Adversarial systems, even when they have incorrect results, can facilitate transmission of incorrect and potentially damaging information. The challenge here is that this type of misuse can impact broad swaths of a population, thereby impacting potentially vulnerable groups as well. This alludes to how the broader impact can be addressed. Natural language processing (and perhaps indeed machine learning in general) is a field in which technology has advanced significantly faster than public policy. Until policy catches up, the onus falls on the technology professionals to ensure to the best of their ability that this technology is developed and deployed in a responsible manner.

References

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. *Computer Science*, arXiv: 1801.04354. Version 5.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. *CoRR*, arXiv:1907.11932. Version 6.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*, arXiv: 1412.6980. Version 1.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards deep learning models resistant to adversarial attacks. *Statistics*, arXiv:1706.06083. Version 4.

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

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Interpretable Adversarial Perturbation in Input Embedding Space for Text. *Computer Science*, arXiv: 1805.02917. Version 1.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *Computer Science*, arXiv:1312.6199. Version 4.

Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021. Natural language adversarial defense through synonym encoding. *Computation and Language*, arXiv:1909.06723. Version 4.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. 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 575–581, New

Orleans, Louisiana. Association for Computational Linguistics.

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.