Option 3

ALBERT Premium for SQuAD 2.0

Stanford CS224N Default Project; mentored by Prerna Dhareshwar

Yanhao Jiang
Department of Electrical Engineering
Stanford University
jiangyh@stanford.edu

Yibing Du
Department of Computer Science
Stanford University
yibingdu@stanford.edu

Abstract

This project aims at achieving a higher performance on the SQuAD 2.0 challenge. Our methods include data preprocessing, improving the ALBERT model, output postprocessing, and ensembling. First, we use data augmentation to increase the size of the training set. Then, we implement several improvements on ALBERT, including incorporating a question answerability classification component, adding extra layers, and tuning the hyperparameters. Finally, after ensembling these models, we use named entity recognition to award the predicted answers with types of words corresponding to the question with a higher probability. Our proposed method reaches the EM/F1 scores of $79.949/82.584$ on the test PCE-division leaderboard.

1 Introduction

Question answering is among the most popular tasks in natural language processing since it is applied in a variety of real-life scenarios. While human readers can usually reach fairly high accuracy when they extract information from paragraphs, it is much more difficult for machine comprehension models to achieve the same effect on profoundly challenging and interesting datasets like SQuAD 2.0 that poses a wide variety questions with some of them being unanswerable in the first place.
Recently, several variations of ALBERT have become the state-of-the-art models on SQuAD 2.0. Inspired major trend is attempting to boost the accuracy of answerability prediction by using verifiers,$later$ we train another ALBERT model as an answerability classifier and take its result into account in our base ALBERT model. Meanwhile, by observing the prediction results on the Explore SQuAD 2.0 page, we propose a unique answer postprocessing mechanism based on named entity recognition, in the hope of outperforming the regular ALBERT model.

2 Related Work

- **BERT**: BERT (Bidirectional Encoder Representations from Transformers) has led to a series of breakthroughs in language representation learning. BERT utilizes the bidirectional training of Transformer, a popular attention model, to train on large corpus. It uses the novel technique of Masked LM (MLM) to achieve this bidirectional training.[1] Shallow output layers can then be fine-tuned on top of the powerful pre-trained weights for different downstream tasks.[2]
- **ALBERT**: While BERT uses a very large network to achieve state-of-the-art performances in many downstream tasks, researchers begin to realize that model size cannot be simply increased boundlessly. It has been shown that problems like hardware memory limitations, training speed, and worsened performance would arise as the model grow larger.
  A more advanced ALBERT model addresses foregoing scalability problem of BERT from two major parameter reduction techniques: factorized parameter reduction and cross-layer parameter sharing. Additionally, it also proposes a self-supervised loss for sentence-order prediction (SOP), which improves upon the BERT version of determining inter-sentence

coherence. Overall, the new ALBERT model improves BERT performance while using much fewer parameters.[3]

- **Limitation of two papers**: The BERT and ALBERT papers did not explore more complex output layer structure for their downstream tasks. Instead, they stick with one simple linear output layer. Furthermore, different data preprocessing and postprocessing techniques are not explored in papers, which can be useful depending on the specific dataset.

3 Approach

3.1 Baseline

The baselines for this projects are the default BiDAF [4] and the BERT base model. For the base BERT model, we are using exactly same set-up and procedures as our main approach for ALBERT below.

3.2 ALBERT base model

Our main approach is based on ALBERT. We imported the pre-trained ALBERT weights and fine-tuned them on top of the question-answering dataset SQuAD 2.0 using a structure similar to the original BERT paper[2]. Specifically, we would concatenate the question (embedded A) and passage (embedded B) together and use them one single input. These two input would be separated by the '[SEP]' token as what specified in the BERT paper. There would be two additional vectors, $S \in R^H$ and $E \in R^H$, being outputted to produce the probability of a word being start or end of the answer span. For example, for a word T_i, the probability that it is the start of a answer is its dot product with start vector S followed by the softmax across all other words in the passage: P(T_i as the start) = $\frac{e^{S \cdot T_i}}{\Sigma_j e^{S \cdot T_j}}$. Similarly, the probability that T_i is the end is P(T_i as the end) = $\frac{e^{E \cdot T_i}}{\Sigma_j e^{E \cdot T_j}}$.
The final answer prediction is defined as the text span where the sum of start and end probability is the highest. Moreover, in terms of the "no-answer" scenario added to the SQuAD 2.0, the prediction is when it makes $S \cdot [\text{CLS}] + E \cdot [\text{CLS}]$ is the highest, where "[CLS]" is the start token added by BERT convention with no specific meaning.
For the loss function, we use cross-entropy loss with mean reduction to take the average inside each batch: loss = $-logp_{start}(i) - logp_{end}(j)$, where i and j in the equation represent the true start and end locations, and $p_{start}(i)$ and $p_{end}(j)$ represent the predicted start and end probability in token i and j, respectively.
Our ALBERT implementation was adapted from Huggingface library example run_SQuAD.py [5].

3.3 Binary Classification

Besides the typical question-answering structure mentioned above, we also added an additional ALBERT-based binary classifier to predict whether a question is answerable or not based on the context. [3] A binary classification task is much easier than the standard question-answering task, and therefore should have higher accuracy identifying "no answer" cases. We used this model as an additional reference to the regular answer span detection task to increase the overall accuracy.
The model we used is the original ALBERT base model with an additional output layer producing 2 logits, one representing is_impossible = FALSE (0), the other one is representing is_impossible = TRUE (1).
As the original BERT model described, we used the final hidden state of the first "[CLS]" token of each Context+Question input as the aggregate sequence representation for our binary classification task. [2] A linear layer with a dropout layer is attached after "[CLS]" token's last hidden state to transform it to be size of 2. The loss function for this model is also a cross entropy loss with formulas and same mean reduction method is used to take loss average inside each batches: loss = $-y_i \cdot log(\hat{y_i}) - (1 - y_i) \cdot log(1 - \hat{y_i})$.
In the prediction phase, the output of binary classification (probability of predicting "no answer") helps decide whether to force the question answering model to provide its best answer or simply force the result to be "no answer". We explored various combination of threshold of forcing to have answer and no answer. Based on Huggingface, we implemented the classification training, evaluation and ensemble codes.

3.4 Additional Linear Output Layers

The proposed question answering model structure for the original ALBERT model only used one linear output linear to transform each token's large final hidden states into the size of 2. Since we observed that directly drops hidden size 768 (base model) or 1024 (large model) to 2 seemed a bit too abrupt, we implemented 4 more linear output layers at the end to gradually reduce size into 2. Specifically, the structure is as follows: linear layer with output size 768, linear layer with output size 768, linear layer with output size 384, linear layer with output size 192, and the final linear layer with output size 2.

3.5 Additional Highway Output Layers with Dropout

The performance of modified 5 linear output layers is not ideal. The possible reason would be over-fitting and gradient vanishing due to the deeper neural network structure. To resolve possible gradient vanishing problem and preserve more information from lower levels, we implemented 3 highway structures to replace the original plain 5 linear layers structure. Additionally, to reduce the the possible over-fitting, we introduced 3 drop-out layers in between each highway structure.
The detailed structure flow is as follows: a highway unit with output size 768, a drop-out layer with probability 0.1, a highway unit with output size 768, a drop-out layer with probability 0.1, a highway unit with output size 768, a drop-out layer with probability 0.1, and a final linear layer with output size 2.
For each Highway unit, the output is obtained by combining the projection with gate: $x_{output} = x_{gate} \odot x_{proj} + (1 - x_{gate}) \odot x_{input}$ where x_{gate} and x_{proj} are calculated as $x_{proj} = ReLU(W_{proj}x_{input}), x_{gate} = \sigma(W_{gate}x_{input})$.

3.6 Data Augmentation (DA)

Inspired by a past CS 224N paper [6], we used Easy Data Augmentation (EDA) [7] to generate similar expressions to the original contexts in the training set so that our model can be trained on an expanded dataset. We adapted the EDA code so that it replaces a certain percentage of terms in the "context" part of the training set and guarantees that the length of each word remains unchanged and words that appear in corresponding answers don't get replaced. Then, we append this changed dataset to the original training set so that we can train on a larger amount of data.

3.7 Named Entity Recognition (NER)

By browsing the SQuAD 2.0 Data Explorer, we realized that different types of questions would expect grammatically different types of answers. Thus, we proposed this original method to postprocess answer candidates so that ones that contain words from the expected categories would have a better chance to selected ultimately. For example, if a question contains "where", we will assign a higher probability to answers that contain location information. We used two different NER identification systems, CoreNLP [8] and NeuroNER [9], because they can identify different types of words. We categorized questions and mapped the word types to question types as shown in Table 1.

QuestionType	Keywords	Count	NeuroNER	CoreNLP
Person	'Who'	638	person	person, title
Time	'When'	441	-	date, duration, set, time
Date	'On what date'	3	-	date
Time (other)	'what decade/century/year'	140	-	date, duration, set, time
Location	'Where'	233	location	location, country, city
City	'Which/W(w)hat city'	14	location	location, city
State	'Which/W(w)hat state'	8	location	location, state_or_province
Country	'Which/W(w)hat country'	38	location	nationality, location, country
Percentage	'percent(age)'	28	-	number, percent
Duration	'How long'	33	-	date, duration, number, set
Money	'How much'	48	-	number, money
Count	'How many'	246	-	number

Table 1: Named Entity Recognition question/answer types, using NeuroNER and CoreNLP.

3.8 Ensemble

We ensembled our models by taking the "votes" from all models with decent performance. Each model will output a file that contains its top candidates along with the corresponding probability. We sum up all of the probabilities for each possible answer string with different assigned weight. The answer with highest vote would be our final prediction.

4 Experiments

4.1 Data

We use default project data-set provided by the instructors, which is adapted from the official SQuAD dataset 2.0 (Training Set v2.0: 40 MB; Dev Set v2.0: 2 MB; Test Set v2.0: 1 MB)[10].

4.2 Evaluation method

The evaluation method for question-answering task is EM and F1 scores, while for the binary classification of question answerability, we use Accuracy, Precision, Recall, and F1 score.

4.3 Experimental details

- **Preprocessing: Data Augmentation**
  Adapted from EDA, we replaced words in the contexts of the training set, excluding those that appeared in corresponding answers or words vital to the sentence meaning or structure (such as "your" or "under"), with a probability of 0.2. Each word is replaced with a random synonym generated by WordNet that has the same length as the original word, so that the starting/ending indices remain unchanged. We preserved the letter cases, symbols, and numbers. Finally, we concatenated it with the original training set and train on this twice-as-large dataset.
- **BiDAF**
  Same as the default configurations, details see reference.[4]
- **BERT and ALBERT**
  We experimented with linear output layers, highway and dropout layers, and binary answerability classification with different learning rate and number of epochs, as shown in Table 2.
- **Postprocessing: Named Entity Recognition**
  We ran CoreNLP and NeuroNER on the dev and test contexts to map each word to its type. Then, for all questions in the corresponding type, we tested several ways to reward the more-likely answers, including adding or multiplying the probabilities of selected candidates.
- **Ensembling**
  We assigned different weights to the output probabilities of all models with relatively satisfying scores to generate the final output.

Model name	pre-trained model	batch size	learning rate	epochs
BERT base	bert-base-cased	6	3e-5	2
ALBERT base	albert-base-v2	6	3e-5	2
ALBERT base + 5 output layers	albert-base-v2	6	3e-5	2
ALBERT base + highway layers	albert-base-v2	6	3e-5	3
ALBERT base classifier 1	albert-base-v2	6	3e-5	2
ALBERT base classifier 2	albert-base-v2	6	3e-5	3
ALBERT base classifier 3	albert-base-v2	6	1e-5	3
ALBERT large classifier	albert-large-v2	3	5e-6	3
ALBERT base + DA (ans changed)	albert-base-v2	6	3e-5	3
ALBERT base + DA (ans unchanged)	albert-base-v2	6	3e-5	3

Table 2: BERT and ALBERT experiments all have max seq length 384 and doc stride 128.

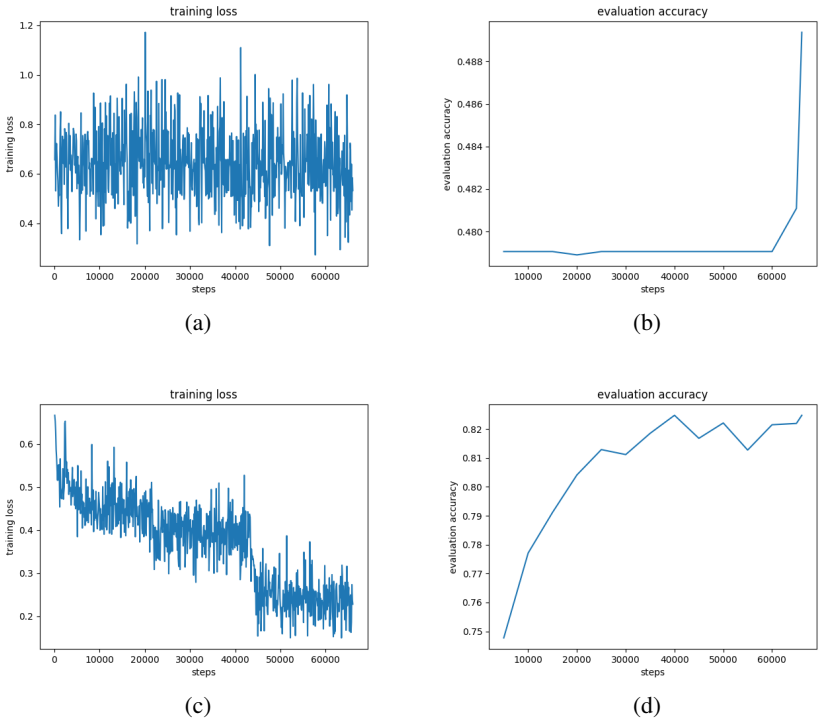
4.4 Results

4.4.1 Binary Classification

With our first classifier model, we obtained accuracy of 0.60, which is relatively low for a binary classification task. After investigating different checkpoints, we observed that till the very late of the training, the model still tends to predict "has answer" around 70% of the time. This indicates an unconverged training because the training data split is in a similar range (66% "has answer").

ALBERT variations	overall accuracy	precision (for 1)	recall (for 1)	F1 (for 1)
① 2 epochs + lr 3e-5	0.60	0.74	0.36	0.48
② 3 epochs + lr 3e-5	0.49	0.59	0.07	0.12
③ 3 epochs + lr 1e-5	0.82	0.85	0.80	0.83
④ 3 epochs + lr 5e-6	0.86	0.87	0.85	0.86

In our second classification model, we increased the training epochs to 3, yet the accuracy drops to 0.49. The oscillated training loss plot and flat evaluation accuracy plot indicate the model isn't really learning (see below figure, (a) and (b)). Suspecting that the original learning rate of 3e-5 was was tuned for question answering task instead of classification, we reduced it to 1e-5 and obtained much better convergence (see below figure, (c) and (d)).



(a) (b)

(c) (d)

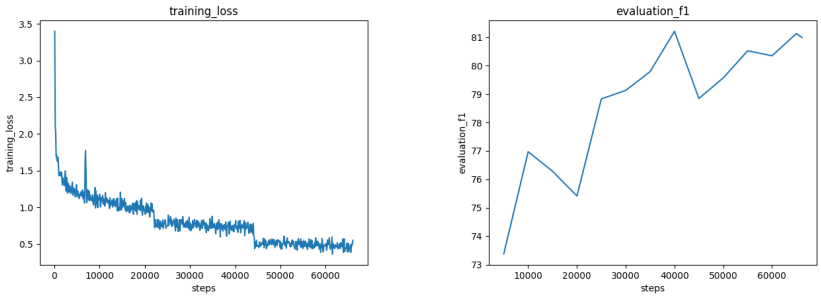
We then further train the classifier on ALBERT large model to achieve better result. The batch size is reduce to 3 due to limited GPU memory and the learning rate is also further reduce to be 5e-6 since the previous training loss is still a bit oscillating. When the probability of "no answer" is high, we force the result as "no answer" and when the probability of "no answer" is low, we try to force the question-answering model to output its best possible answer. We performed grid search on different threshold and obtained results as shown in Table 3.
The best no answer threshold we found is 0.99. However, no matter what the has answer threshold is, it only degrades the result. This is possibly because that after the classifier forces has answer prediction, the model still has to generate a correct answer span. This is a much harder task than simply outputting no answer.

Table 3: Regular ALBERT models and classifiers with and without answer thresholds.

Model name	no ans threshold	has ans threshold	DEV EM	DEV F1
ALBERT base + classifier base	0.7	-	79.48	82.18
ALBERT base + classifier base	0.9	-	79.48	82.35
ALBERT base + classifier base	0.99	-	**79.58**	**82.35**
ALBERT base + classifier base	0.995	-	79.47	82.27
ALBERT base + classifier base	0.7	0.01	78.63	81.51
ALBERT base + classifier base	0.7	0.001	79.48	82.18
ALBERT base + classifier base	0.7	0.0025	79.32	82.05
ALBERT base + classifier large	0.99	-	**80.56**	**83.32**

4.4.2 ALBERT with various additional output layers

Implementing ALBERT with additional 5 linear output layers received degraded EM and F1 scores of 78.35 and 81.52. To resolve possible over-fitting and gradient vanishing due to the deeper neural network structure, we implemented another variation with 3 repeated highway + dropout structures and received EM/F1 scores of 78.30 and 81.21. From the evaluation graph below, we observed that the F1 score never outperformed the regular ALBERT. One possible reason is that ALBERT is already deep enough to learn the QA task and extra output layers only makes its learning harder.



(a) (b)

4.4.3 Data Preprocessing

While we expected to boost the performance with data augmentation, it turns out that it actually harms the scores. Comparing to EM/F1 scores of the fine-tuned ALBERT base model, the scores of the same model trained on a twice-as-large training set decreases significantly, no matter if we preserved all words that appeared in answers from being replaced or not. Since replacing 10% rather than 20% of the words slightly improved the performance, one possible reason is that a relatively large change might have distorted the original context structure.

Data augmentation (DA) usage	DEV EM	DEV F1
Without DA	79.06	81.91
With 20% DA (changed answer words)	76.01	79.60
With 20% DA (preserved answer words)	75.77	79.09
With 10% DA (changed answer words)	76.43	79.88

4.4.4 Answer Postprocessing

While using CoreNLP and multiplying the probability by 1.5 or adding 0.2 on top of the probability achieved relatively good results, the EM/F1 scores were still slightly lower than the regular ensembled result. Since NeuroNER had much fewer tags available, the highest scores it achived after abandoning the 'MISC' tag and adding 0.2 to each probability was still lower than CoreNLP's.

NER	MISC	Calculation	DEV EM	DEV F1
Neither	-	-	80.158	82.612
NeuroNER	Yes	probability $\times$ 1.5 if any match	79.615	82.325
NeuroNER	Yes	probability $\times$ 2 if any match	79.631	82.328
NeuroNER	No	probability $\times$ 2 if any match	79.648	82.358
NeuroNER	No	probability $+0.2$ if any match	**79.664**	**82.375**
CoreNLP	-	probability $\times$ 1.5 if any match	**80.092**	**82.578**
CoreNLP	-	probability $\times$ 2 if any match	79.631	82.360
CoreNLP	-	probability $+0.2$ if any match	**80.059**	**82.510**
CoreNLP	-	probability $\times 1.5$ for each match, $\times 0.9$ for each non-match	**79.681**	**82.334**
both	No	probability $+0.2$ if any match	79.664	82.375

4.4.5 Ensembling

The first ensemble of 5 different models boosted the EM by 1.096 and F1 by 0.702 compared with the highest performance individually. After adding the ALBERT with highway model, the ensemble performance again being boosted by +0.592 (EM) and +0.429 (F1). This is very interesting because ALBERT with highway itself actually performs worse than other models. A possible reason is that ALBERT with highway has a radically different structure with the rest, which might fix mistakes that other more regular ALBERT models make.

Ensemble Method	DEV EM	DEV F1
BERT base + ALBERT base + ALBERT 5 layers + ALBERT base with base classifier + 0.5 BiDAF	80.158	82.612
BERT base + ALBERT base + ALBERT 5 layers + ALBERT base with base classifier + 0.5 BiDAF + ALBERT highway	80.750	83.042
BERT base + ALBERT base + ALBERT 5 layers + ALBERT base with base classifier + 0.5 BiDAF + ALBERT highway + ALBERT with augmentation ①	80.833	83.117
BERT base + ALBERT base + ALBERT 5 layers + ALBERT base with base classifier + 0.2 BiDAF + 1.2 ALBERT highway + ALBERT with augmentation ① + ALBERT with augmentation ②	81.013	83.255
BERT base + ALBERT base + ALBERT 5 layers + ALBERT base with base classifier + 0.7 BiDAF + 1.8 ALBERT highway + ALBERT with augmentation ① + 2.5 ALBERT base with large classifier	81.063	83.499

4.4.6 Summary

On test PCE leaderboard, our best ensemble model reached the scores of **EM: 79.949 F1: 82.584**. Our ensemble scores are shown in the table below. The binary classifier boosted the accuracy of question answerability and a combination of ALBERT with highway layers and ALBERT with linear output layers enhanced the scores even more. While data augmentation and named entity recognition did not work as well as we expected, we still see some potential to adjust the exact implementation of these methods.

Model name	DEV EM	DEV F1
BiDAF	57.86	61.30
BERT base	73.86	77.24
ALBERT base	79.06	81.91
ALBERT base 5 output layers	78.35	81.52
ALBERT base highway layers	78.30	81.21
best ALBERT base + classifier	79.58	82.35
best ALBERT base + classifier large	80.56	83.32
best ALBERT base with augmentation	76.01	79.60
best ensemble	**81.063**	**83.499**

5 Analysis

We decided to qualitatively evaluate our model by examining two false predictions that correspond to two features we implemented.

- **Context** (trimmed): A term used originally in derision, Huguenot has unclear origins. Various hypotheses have been promoted.
- **Question**: The term Huguenot was originally meant to confer?
- **Expected answer**: “derision", “derision", “derision"; **Prediction**: No Answer

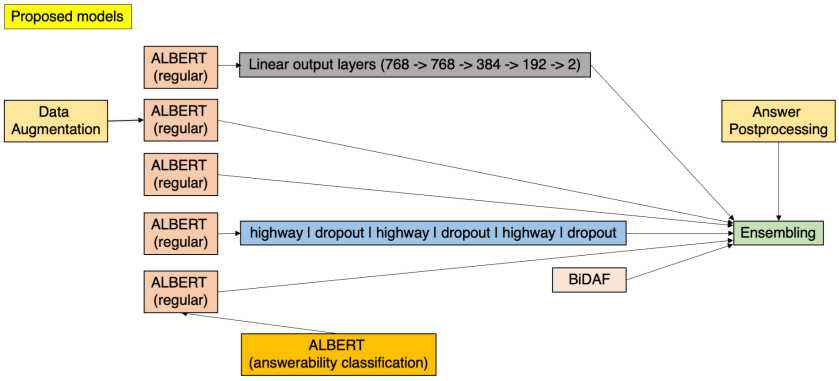
The binary classifier we added to the regular ALBERT boost the accuracy of ruling “no answer" questions as “no answer", yet it is unable to handle answerable questions very well. The original meaning of Hugeunot is hinted in a short phrase “used originally in derision", which is very brief and can be read in other ways, making it likely to be considered a “no answer" question.

- **Context** (trimmed): [...] the Ottoman Empire was a powerful multinational, multilingual empire controlling much of Southeast Europe, Western Asia, the Caucasus, North Africa, and the Horn of Africa.
- **Question**: The Ottoman empire controlled territory on three continents, Africa, Asia and which other?
- **Expected answer**: “Europe", “Europe", “Europe"; **Prediction**: “Southeast Europe, Western Asia, the Caucasus,"

While the question asks about another continent, the predicted answer includes several specific regions because our model fails to tell what kind of word is expected. Though our NER postprocessing sometimes harms accuracy, this problem might be solved by implementing fine-grained NER techniques that specify the expected answer types.

6 Conclusion

The model we proposed, visualized by the following graph, consists of three parts. First, we added output layers for ALBERT, including linear and highway-dropout so that the large final hidden size would not be dropped too immediately. Secondly, we implemented an ALBERT-based binary answerability classification to increase the accuracy of predicting “No Answer". Thirdly, we trained our model on a larger dataset after using data augmentation to replace words in the training contexts with their synonyms. Fourthly, we proposed the unique idea of using named entity recognition to postprocess the candidate answers so that answers containing the correct types of words are assigned higher chance. Finally, we improved ALBERT performance on SQuAD 2.0 significantly by ensembling these models.
Later on, we plan to adjust the percentage of changed words for data augmentation and experiment with different question/keyword types for named entity recognition. Moreover, we plan to boost the prediction accuracy of answerable questions by tuning the way we combine the results of the regular ALBERT and the binary classifier.



References

[1] Bert explained: State of the art language model for nlp. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. Accessed: 2020-03-02.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv*, 2019.

[3] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *Association for Computational Linguistics (ACL)*, 2020.

[4] Stanford. Cs 224n default final project: Question answering on squad 2.0. 2020.

[5] Huggingface tranformers. https://github.com/huggingface/transformers. Accessed: 2020-03-02.

[6] Wen Zhou, Xianzhe Zhang, and Hang Jiang. Ensemble bert with data augmentation and linguistic knowledge on squad 2.0. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15845024.pdf. Accessed: 2020-03-16.

[7] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *arXiv*, 2019.

[8] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[9] Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2017.

[10] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.