Gestalt: a Stacking Ensemble for SQuAD2.0

Stanford CS224N Default Project. **Please grade!**

Mohamed El-Geish
Department of Computer Science
Stanford University
elgeish@stanford.edu

Abstract

We propose a deep-learning system — for the SQuAD2.0 task — that finds, or indicates the lack of, a correct answer to a question in a context paragraph. Our goal is to learn an ensemble of heterogeneous SQuAD2.0 models that, when blended properly, outperforms the best model in the ensemble per se. We created a stacking ensemble that combines top-N predictions from two models, based on ALBERT and RoBERTa, into a multiclass classification task to pick the best answer out of their predictions. We explored various ensemble configurations, input representations, and model architectures. For evaluation, we examined test-set EM and F1 scores; our best-performing ensemble incorporated a CNN-based meta-model and scored 87.117 and 90.306, respectively — a relative improvement of 0.55% for EM and 0.61% for F1 scores, compared to the baseline performance of the best model in the ensemble, an ALBERT-based model, at 86.644 EM and 89.760 F1 scores.

1 Introduction

Despite the simplicity of the idea, ensemble learning has been widely successful in a plethora of tasks — ranging from machine learning contests to real-world applications [1]. We aim at using ensemble learning to create a deep-learning system for a machine reading comprehension application: question answering (QA). The main goal of QA systems is to answer a question that's posed in a natural manner using a human language. In this research area, we find many examples for variants of the QA task [2]. Closed-domain QA systems answer questions about a specific domain while open-domain ones answer questions about a myriad of topics [3, 4]. Knowledge-base QA systems find answers in a specific knowledge base such as Freebase [5]. Multiple-choice QA systems pick an answer out of multiple choices like in the MCTest [6] and RACE [7] tasks. Some QA systems directly answer a given question by generating complete sentences like in ELI5 [8] while others extract a short span of text in a corresponding context paragraph to present it as an answer; the latter is the main objective of SQuAD (the Stanford Question Answering Dataset) models [9]. In SQuAD2.0, an additional challenge was introduced: The model has to indicate when a question is unanswerable given the corresponding context paragraph — see Figure 1 for SQuAD2.0 examples.

SQuAD2.0 systems face many challenges: The task requires accurately concocting some forms of natural language representation and understanding that aid in processing a question and the context to which it relates, then selecting a reasonable correct answer that humans may find satisfactory or indicate the lack of such answer. The vast majority of modern systems, which outperform humans according to the SQuAD2.0 leaderboard [10], try to find two indices: the start and end positions of the answer span in the corresponding context paragraph (or sentinel values if no answer was found); this is usually done with the aid of Pre-trained Contextual Embeddings (PCE) models, which help with language representation and understanding [11, 12, 13, 14, 15, 16]. The SQuAD2.0 leaderboard shows that ensembles improve upon the performance of single models: In [11], BERT introduced an ensemble of six models that added 1.4 F1 points; ALBERT's ensemble averages the scores of 6 to 17 models, leading to a gain of 1.3 F1 points, compared to the single model [17]; RoBERTa and XLNet also introduced ensembles but did not provide sufficient details [14, 12]. Our QA ensemble system

Paragraph: "Bethencourt took the title of King of the Canary Islands, as vassal to Henry III of Castile. In 1418, Jean's nephew Maciot de Bethencourt sold the rights to the islands to Enrique Pérez de Guzmán, 2nd Count de Niebla."

Question 1: "What title did Henry II take in the Canary Island?"
Ground Truth Answer: <No Answer>
Plausible Prediction: King of the Canary Islands

Question 2: "Who bought the rights?"
Ground Truth Answer: Enrique Pérez de Guzmán
Plausible Prediction: Pérez

Figure 1: Two examples of SQuAD2.0 questions and answers written by crowdworkers, along with plausible model predictions: the one in red is incorrect while the one in blue is incomplete.

added a gain of 0.473 EM points (0.55% relative gain) and 0.546 F1 points (0.61% relative gain), compared to the best-performing single model in the ensemble, when measured using the project's test set (which is different from the official SQuAD2.0 hidden test set).

A key difference in our approach is the use of stacking [18] to combine top-N predictions produced by each model in the ensemble. We picked heterogeneous PCE models, fine-tuned them for the SQuAD2.0 task, and combined their top-N predictions in a multiclass classification task using a convolutional neural meta-model that selects the best possible prediction as the ensemble's output. Since each model in the ensemble is learned in an idiosyncratic manner, we expect their results (given the same input) to vary — a behavior that's analogous to asking humans, who come from diverse backgrounds, for their opinions.

2 Related Work

Pre-trained Contextual Embeddings (PCE) models have been instrumental in advancing language representation learning and achieving state-of-the-art results in many NLP tasks; the Bidirectional Encoder Representations from Transformers (BERT) family of models have been at the forefront [11]. ALBERT, A Lite BERT, has been notably excelling at the SQuAD2.0 task [16] — especially when used in an ensemble: At the time of writing, the top 7 models on the SQuAD2.0 leaderboard are ALBERT-based ensembles [10]. In [16], the ALBERT ensemble selects model checkpoints with the best development set performance then averages their prediction scores to produce an aggregate answer for a SQuAD2.0 question (or to indicate the lack of an answer). ALBERT's ensemble achieved an F1 score of 92.2 while the single-model result was 90.9 (after 1.5M training steps).

One challenge of averaging prediction scores is giving each model in the ensemble an equal voting power regardless of its performance; another is susceptibility to outliers — a single outlier can tip the scales easily. There are more sophisticated ways of combining models to address such challenges, which we listed in Appendix B, including assigning each vote a weight; we may consider the aforementioned methods special cases of stacked generalization [18]. Stacking (another name for stacked generalization) works by learning a meta-model that combines the predictions of various models in order to produce predictions that minimizes the generalization error. A stacking ensemble exploits the independence of heterogeneous models, which were built differently, since the probability of them colluding to give wrong answers is minimal [19].

Stacking ensembles for SQuAD2.0 are to benefit from blending heterogeneous PCE models like XLNet [12], DistilBERT [13], RoBERTa [14], Text-To-Text Transfer Transformer (T5) [15], and ALBERT [16] — each of which has its own unique contributions. While BERT learns using Masked Language Modeling and Next Sentence Prediction (NPS) loss, XLNet maximizes the expected likelihood over all permutations of words in a sentence (Permutation Language Modeling). DistilBERT is a distilled version of BERT that's cheaper and faster to fine-tune while retaining 97% of its language understanding capabilities. RoBERTa improved upon BERT's performance thanks to careful hyperparameter choices, longer training using more data that include longer sequences, and removing the NPS objective. T5 scaled up model size (up to 11B parameters) and learned from 120B words — the Colossal Clean Crawled Corpus (C4) dataset it introduced in [15]. ALBERT improved upon BERT in terms of parameter efficiency and the use of inter-sentence coherence loss instead of NPS loss.

In the same manner ensembles showed significant improvements in performance for tasks in [18, 1] (e.g., the NETtalk task of translating text to phonemes), we propose approaches below to create a stacking ensemble for the SQuAD2.0 task with the performance improvement goal in mind.

3 Approaches

Creating a stacking ensemble for SQuAD2.0 entails building a pipeline of two stacked stages: level 0 and level 1. In level 0, models learn from SQuAD2.0 and produce predictions, which are then used as input to level 1, for a meta-model, to produce better predictions. We extend this approach further by producing top-N hypotheses from each level-0 model in the ensemble to feed as input to level 1; as we show in Figure 4, the set of top-N predictions (when $N > 1$, compared to the set of top-1 predictions) has a much better chance of including the correct answer.

To learn level-0 models, we fine-tuned various state-of-the-art PCE models using the provided SQuAD2.0 starter files [20], to follow proper data hygiene practices given the dev-test split. To learn the meta-model in level 1, we gave it a classification task: We selected the top-N hypotheses produced by M level-0 models, computed the F1 score distribution for the resulting $M \times N$ hypotheses given the ground-truth answers, and then asked the meta-model to predict the F1 score distribution for a set of $M \times N$ hypotheses; the ensemble's predicted answer is the argmax of the predicted F1 scores.

For level 0, we selected the top-8 hypotheses produced by two models: `albert-xxlarge-v1` and `roberta-base` [21]. The former is the best performing level-0 model (on the dev set) and doubles as the baseline: It achieved an EM score of 85.966 and an F1 score of 88.945 while its top-8 scores were 94.965 (EM) and 95.473 (F1). We picked `roberta-base` because it was the best performing model outside the ALBERT family of models [1] and because of its relatively smaller size (125M parameters, compared to 223M parameters in `albert-xxlarge-v1`); notably, its top-1 scores were relatively low: 75.337 (EM) and 78.683 (F1); however, its top-8 scores were 94.373 (EM) and 95.372 (F1). More importantly, combining the two set of hypotheses yielded the synergistic scores of 98.289 (EM) and 98.539 (F1), thanks to the complementary nature of the blend, which we can use as an oracle for the meta-model in level 1. See Table 1 for a detailed comparison of PCE models' scores.

For level 1, 16 hypotheses $\boldsymbol{h}$ (top-8 from two models) and their respective F1 scores $s_i \in [0, 1]$ (with respect to the ground truth) were interleaved to make up a vector which preserved their original relative order. When a model produced fewer than 8 hypotheses, they were padded with a special token (`<ap>`) and a respective F1 score of 0; the percentages of padded examples in the train, dev, and test datasets were 10.9%, 0%, and 11.4%, respectively. The no-answer hypothesis also gets its own special token (`<na>`). We prepended a unique prefix token to each of the 16 hypotheses to identify their positions (`<h1>, <h2>, ..., <h16>`), then encoded each input string using a T5 tokenizer [15, 22], which was reconfigured to learn about the newly added special tokens, and obtained 16 vectors of token identifiers $\boldsymbol{t}$. Each $\boldsymbol{t}$ vector is of size 16; tokens were truncated if they exceeded the vector's size (the maximum length for an answer was set to 30 in level 0); conversely, tokens were padded with a padding token (`<pad>`) if they were fewer than 16. We concatenated the 16 vectors to represent the input hypotheses to the meta-model as an input vector $\boldsymbol{x}$, The target $\boldsymbol{y}$ was designed to be the distribution of scores $\boldsymbol{s}$ after applying the $\mathrm{softmax}$ function. See Figure 2 for an illustration.

Our best-performing meta-model used `t5-small` [21, 15] to generate contextual embeddings, each is of size 512, for tokens in $\boldsymbol{x}$. The PCE model in level 1 (in this case, `t5-small`), was not fine-tuned for any specific task before using it as a meta-model module; we allowed the PCE model's parameters to change given the new task and special tokens we introduced in level 1; we also resized the PCE model to include said tokens in its vocabulary. T5 was our default choice for a PCE model in level 1 given the flexibility it provides for arbitrary special tokens, state-of-the-art performance at various NLP tasks, and being different from the two PCE models we fine-tuned in level 0: `albert-xxlarge-v1` and `roberta-base`; empirical results, discussed in Section 4, supported our default choice.

[1]XLNet's results (see Table 1) were ignored because they were too low, possibly due to a bug in the transformers library's implementation: `https://github.com/huggingface/transformers/issues/2651`

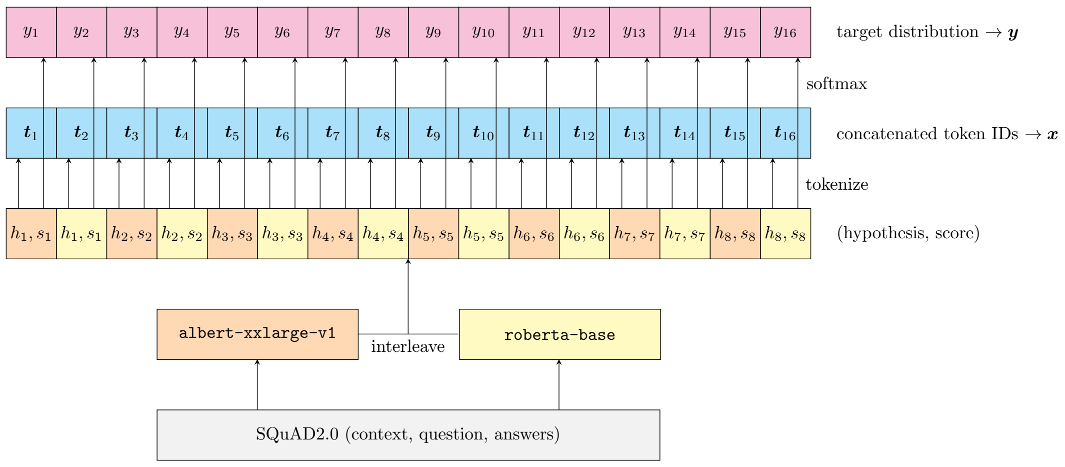

y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12} y_{13} y_{14} y_{15} y_{16}
target distribution $\rightarrow \boldsymbol{y}$
softmax
$\boldsymbol{t}_1$ $\boldsymbol{t}_2$ $\boldsymbol{t}_3$ $\boldsymbol{t}_4$ $\boldsymbol{t}_5$ $\boldsymbol{t}_6$ $\boldsymbol{t}_7$ $\boldsymbol{t}_8$ $\boldsymbol{t}_9$ $\boldsymbol{t}_{10}$ $\boldsymbol{t}_{11}$ $\boldsymbol{t}_{12}$ $\boldsymbol{t}_{13}$ $\boldsymbol{t}_{14}$ $\boldsymbol{t}_{15}$ $\boldsymbol{t}_{16}$
concatenated token IDs $\rightarrow \boldsymbol{x}$
tokenize
h_1, s_1 h_1, s_1 h_2, s_2 h_2, s_2 h_3, s_3 h_3, s_3 h_4, s_4 h_4, s_4 h_5, s_5 h_5, s_5 h_6, s_6 h_6, s_6 h_7, s_7 h_7, s_7 h_8, s_8 h_8, s_8
(hypothesis, score)
albert-xxlarge-v1
interleave
roberta-base
SQuAD2.0 (context, question, answers)

Figure 2: A simplified flow of input $\boldsymbol{x}$ and output $\boldsymbol{y}$ data processing to learn a meta-model.

The best-performing meta-model architecture mapped the embeddings matrix to the target distribution $\boldsymbol{y}$ using a series of transformations: transposing dimensions so that the embedding size is the number of input channels for a convolution; two convolutional blocks, each applied a 1D convolution then max-pooling, with 1024 and 64 output channels, respectively; followed by two fully connected layers, with 64 and 16 nodes, respectively; and finally a log-softmax output layer, which predicts the log-probabilities $\hat{\boldsymbol{y}}$ for the 16 input hypotheses. We picked the Kullback–Leibler (KL) divergence loss [23] with a summative reduction as the cost function to minimize:

$$D_{\mathrm{KL}}(\hat{\boldsymbol{y}} \parallel \boldsymbol{y}) = \sum_i \hat{y}_i \log \frac{\hat{y}_i}{y_i}$$

The KL divergence loss impels the meta-model to learn to predict log-probability scores for a mix of — potentially multiple — correct, partially correct, and incorrect hypotheses in the input $\boldsymbol{x}$. We used the rectified linear unit (ReLU) activation for nonlinearity [24]. For optimization, we picked Adam with a dynamic learning rate that gets reduced when the dev F1 score plateaus [25]. We implemented the ensemble using PyTorch, parts of the provided starter code, and PCE models from the transformers library [26, 20, 22].

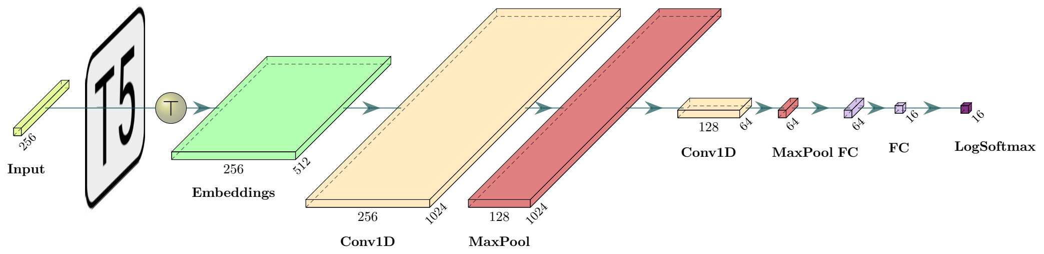


Figure 3: Architecture of the best-performing meta-model in level 1.

4 Experiments

4.1 Evaluation Methods

To evaluate the ensemble, we used the same evaluation metrics as the SQuAD2.0 task: Exact Match (EM) and F1 scores.

- **EM:** is a binary measure of correctness — the score is 1 if the predicted answer exactly matches the ground truth; otherwise it's 0.

- **F1:** is the harmonic mean of precision and recall of words in the predicted answer given the ground truth. $F1 = 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$

Predictions and ground-truth answers are normalized before evaluation: We convert them to lowercase and remove punctuation, articles, and extraneous whitespaces. For each question in the evaluation dataset, multiple ground-truth answers are supplied; we pick the best score for a prediction given the ground-truth answers to the respective question, then average the scores across the entire dataset. We use the performance, EM and F1 scores, of the best model in the ensemble as a baseline to outperform.

To evaluate the performance of level-0 models and their respective top-N predictions, we used the same metrics listed above; we evaluated the models at $N \in \{1, 2, 4, 8, 16, 32\}$ and picked the best score out of the N results for each metric. We used the results of level 0 to curate the input to level 1 (the choice of models and the value of N); see Figure 4 for a comparison of the results at a glance.

4.2 Data

For level 0, we used the given SQuAD2.0 starter files [20] (with half of the official dev dataset repurposed as the test dataset) to fine-tune PCE models. The train/dev/test splits consist of 129,941/6,078/5,915 examples, respectively. Each input consists of a question and a context paragraph pair; each output represents the span of text in the context paragraph that answers the respective question or indicates the lack of such answer. Almost half of the questions in the dataset are unanswerable given their respective contexts [27].

For level 1, we selected two models (`albert-xxlarge` and `roberta-base`) from level 0 to contribute to the ensemble; we used the top-8 predictions and their respective F1 scores for the train, dev, and test datasets from level 0 to create the respective train, dev, and test datasets required to learn the neural meta-model in level 1. We ignored the targets (answers in level 0 and F1 score distribution in level 1) for the test data sets.

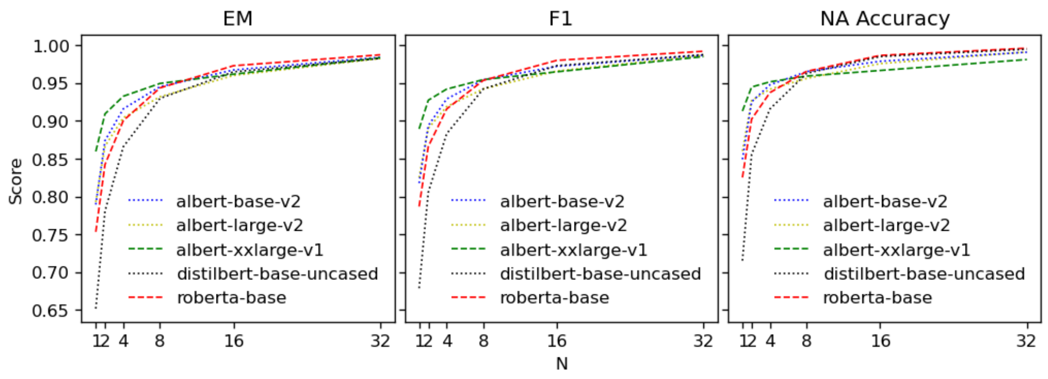


Figure 4: Top-N EM, F1, and No-Answer Accuracy scores of level-0 models. Using the top-8 answers as input to the meta-model provides a good lift in metrics (at a relatively small N).

4.3 Experimental Details

4.3.1 Fine-Tuning Level-0 PCE Models

Table 1 shows detailed configurations and results of fine-tuning level-0 PCE models using SQuAD2.0. XLNet models were not considered for the ensemble because they scored too low, possibly due to a bug in the transformers library's implementation of XLNet [28]. We used `albert-xxlarge-v1` since it outperforms v2, thanks to carefully tuned hyperparameters [17]. The performance of our `albert-xxlarge-v1` model (85.97 EM and 88.95 F1 on our dev set; 86.64 EM and 89.76 F1 on our test set) is slightly worse than what's reported in [16] and the official SQuAD2.0 leaderboard (87.4 EM and 90.2 F1 on the official dev set; 88.107 EM and 90.902 F1 on the official test set). The discrepancies can be explained by the differences in the evaluation datasets, almost halving the dev set used for model selection, and under-training (due to the limited training budget).

	xlnet-base-cased	xlnet-large-cased	distilbert-base-uncased	roberta-base	albert-base-v2	albert-large-v2	albert-xxlarge-v1
Gradient Accumulation Steps	1	24	24	24	24	1	24
Learning Rate	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5
Max Answer Length	30	30	30	30	30	30	30
Max Query Length	64	64	64	64	64	64	64
Max Sequence Length	384	384	384	384	384	384	512
Training Epochs	4	3	4	4	3	5	4
Training Batch Size	16	4	32	16	8	8	1
Top-1 Dev EM Score	35.604	38.549	65.186	75.337	78.957	79.286	85.966
Top-1 Dev F1 Score	40.339	42.152	67.890	78.683	81.789	82.525	88.945
Top-8 Dev EM Score	—	—	92.991	94.373	94.538	93.205	94.965
Top-8 Dev F1 Score	—	—	94.245	95.372	95.447	94.230	95.473

Table 1: Fine-tuning experiments and notable results for level-0 PCE models.

4.4 Level-1 Meta-Model

We explored various data representations, embedding models, meta-model architectures, and other hyperparameters in order to improve performance. In designing experiments, we examined the following variables:

Representation design choices. (i) assigning each hypothesis in the input its own unique special token as a prefix vs. using a single separator token (ii) tokenizing and padding each hypothesis and token pair separately vs. concatenating the pairs then tokenizing with padding at the end (iii) PCE models for language representation: `t5-small`, `t5-base`, `xlnet-large-cased`, and `albert-xlarge-v2` (iv) adding the question to the input with its own unique prefix token (v) maximum answer length values of 16, 24, and 32; the latter doesn't truncate any input since the maximum length expected is 31: 30 from the output of level-0 models and the prefix token (vi) penalizing the model further when it erroneously picks an answer for an unanswerable question or a no-answer hypothesis for an answerable one; incorrect hypotheses, with respect to the answerability condition of the given question, are assigned scores of -1 instead of their respective F1 scores; we refer to this variant below in Table 2 as biased $\boldsymbol{y}$; see Figure 8 for an example.

Architecture design choices. To transform the embeddings, we started with a single fully connected hidden layer then increased the complexity of the model by adding more nodes and layers (up to 4) as needed. We also experimented with 1D convolutions (up to 3 blocks) followed by fully connected layers (up to 2) in the same manner. In addition, we explored the addition of a self-attention layer (that attends to the embeddings) to the model variants above.

Regularization techniques. (i) dropout for hidden layers, that transform the embeddings, with hand-tuned values for the dropout probability $p \in \{0.1, 0.2, 0.3, 0.5, 0.6, 0.8\}$ (ii) dropout for the embeddings with $p = 0.2$ (iii) adding Gaussian noise, with $\sigma = 0.1$, to the embeddings

Optimization design choices. We tested multiple values for Adam's initial learning rate: 0.01, 0.001, 0.0005, and 0.0001. For the reduce-on-plateau scheduler, we experiments with 10 and 3 as values for the patience parameter.

Given the large number of combinations, we tested variables with the highest expected impact first, one at a time. Then, we used the newly obtained best-performing configuration as a baseline for the next round of experiments. To mitigate myopic design choices, certain variants (e.g., convolutional and non-convolutional architectures) were retested with addition or ablation of other significant variables. It's worth noting that EM and F1 dev scores moved in the same direction (increasing or decreasing together) but not necessarily with the same ratio; when a trade-off was needed for model selection, we picked variants that better improved the F1 score.

4.5 Results

On the PCE leaderboard, we reported the following test scores:

Model	**Test EM Score**	**Test F1 Score**
Baseline (`albert-xxlarge-v1`)	86.644	89.760
`t5-small`, 0.2 dropout, 2 FC layers (256 $\rightarrow$ 16), biased $\boldsymbol{y}$	87.050	90.239
Best-performing ensemble (described in Section 3)	**87.117**	**90.306**

Table 2: Test scores we reported on the PCE leaderboard.

Given the use of `albert-xxlarge-v1` scores of the level-1 oracle (the combined output of level-0 hypotheses) of 98.289 (EM) and 98.539 (F1), we expected gains comparable to the ones obtained by ALBERT's ensemble (1.3 F1 points). Error analysis of output samples shows that the predominant error class is answering an unanswerable question; this could be explained by the design choices of the target distribution $\boldsymbol{y}$ and the KL divergence loss function. A potential improvement here might be obtained by breaking down the level-1 meta-model into a pipeline of two tasks: binary classification (answerable vs. unanswerable) and multiclass classification (which answer to output) when the first stage indicated that the question is answerable. Alternatively, the two tasks can be combined in a multi-objective optimization setting.

5 Analysis

Using the dev set, we found 162 disagreements (out of 6,078 examples) between the normalized answers of the ensemble and the baseline model; 73% of the disagreements were attributed to a difference in opinion regarding the answerability of questions. In addition, we manually inspected 50 randomly chosen results from the test set (without answers) to better understand the ensemble's output, compared to the baseline. We also repeated the same analyses for two variants of the ensemble to include a qualitative factor into the model selection process. Given our own on-the-fly answers to questions in the random test sample, we found that the best-performing ensemble, unsurprisingly, made the fewest mistakes and when it disagreed with the baseline model, other ensemble variants agreed.

Below, we list a few examples of disagreements between the best-performing ensemble and the baseline model.

Paragraph: "The further decline of Byzantine state-of-affairs paved the road to a third attack in 1185, when a large Norman army invaded Dyrrachium, owing to the betrayal of high Byzantine officials. Some time later, Dyrrachium—one of the most important naval bases of the Adriatic—fell again to Byzantine hands."

Question 1: "Who betrayed the Normans?"
Ground Truth Answer: <No Answer>
Baseline's Answer: high Byzantine officials
Ensemble's Answer: <No Answer>

Figure 5: An example of the baseline model incorrectly answering an unanswerable question while the best-performing ensemble picked the correct no-answer hypothesis, which was ranked 2nd and 3rd by the baseline model (`albert-xxlarge`) and `roberta-base`, respectively, in level 0.

Paragraph: "If the input size is n, the time taken can be expressed as a function of n. Since the time taken on different inputs of the same size can be different, the worst-case time complexity T(n) is defined to be the maximum time taken over all inputs of size n..."

Question 1: "How is worst-case time complexity written as an expression?"
Ground Truth Answer: T(n)
Baseline's Answer: T(n)
Ensemble's Answer: <No Answer>

Figure 6: An example of the best-performing ensemble incorrectly picking a no-answer hypothesis, which was ranked as the top answer by `roberta-base` in level 0. It's worth noting that 7 hypotheses, out of the 16 in level 0, had an F1 score of 0; that's expected given the very short answer and answers that don't overlap with the ground truth.

Paragraph: "The most commonly used reduction is a polynomial-time reduction. This means that the reduction process takes polynomial time. For example, the problem of squaring an integer can be reduced to the problem of multiplying two integers. This means an algorithm for multiplying two integers can be used to square an integer. Indeed, this can be done by giving the same input to both inputs of the multiplication algorithm. Thus we see that squaring is not more difficult than multiplication, since squaring can be reduced to multiplication."

Question 1: "According to polynomial time reduction squaring can ultimately be logically reduced to what?"
Ground Truth Answer: multiplication
Baseline's Answer: multiplication
Ensemble's Answer: multiplication, since squaring can be reduced to multiplication.

Figure 7: An example of the best-performing ensemble picking a reasonable answer; yet, it's still penalized given the ground-truth answers — all agreed on the single-word answer, which is expected given the preference of short answers over long ones in SQuAD2.0. The ensemble ignored the two top answers, which were correct, in level 0.

6 Conclusion

Ensemble learning is a tremendously useful technique to improve upon state-of-the-art models; it helps models generalize better and overcome their weaknesses. In a stacking ensemble settings, heterogeneous level-0 models can complement each other like a gestalt; when blended properly, the ensemble outperforms the best model in level 0. Our best-performing ensemble combined `albert-xxlarge-v1` and `roberta-base` in level 0; incorporated `t5-small` to generate contextual embeddings; transformed the embeddings through a CNN-based meta-model; and achieved relative improvements of 0.55% for EM and 0.61% for F1 scores, compared to the baseline performance of the best model in level 0. We believe there's still room for improvement and future work that explores experiments such as backpropagating errors to level-0 models while training the meta-model in level 1; testing various combinations of level-0 models and values of $N > 8$ which represent better input to level 1; data augmentation using distractors and incorrect answers; multi-objective optimization that optimizes for a linear combination of both EM and F1 scores; and a thorough hyperparameter optimization. Finally, we conclude that stacking generalization can benefit state-of-the-art SQuAD2.0 models to outperform their single-model counterparts and can improve upon existing systems, ensembles and single models alike.

References

[1] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[2] J. Eisenstein. *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning series. MIT Press, 2019.

[3] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[4] David Ferrucci. Build watson: an overview of deepqa for the jeopardy! challenge. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 1–2, 2010.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[6] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[7] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[8] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy, July 2019. Association for Computational Linguistics.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 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, November 2016. Association for Computational Linguistics.

[10] The stanford question answering dataset. `https://rajpurkar.github.io/SQuAD-explorer/`. Accessed: 2020-03-15.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics.

[12] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[16] 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 *International Conference on Learning Representations*, 2020.

[17] google-research/albert. `https://github.com/google-research/ALBERT`. Accessed: 2020-03-03.

[18] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[19] I.H. Witten, E. Frank, M.A. Hall, and C.J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*, page 497. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2016.

[20] Starter code for stanford cs224n default final project on squad 2.0. `https://github.com/minggg/squad`. Accessed: 2020-03-15.

[21] Pretrained models. `https://huggingface.co/transformers/pretrained_models.html`. Accessed: 2020-03-03.

[22] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[23] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[24] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

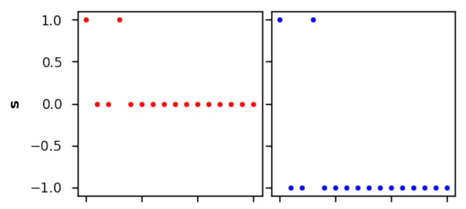
[27] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[28] Xlnet squad2.0 fine-tuning - what may have changed? `https://github.com/huggingface/transformers/issues/2651`. Accessed: 2020-03-03.

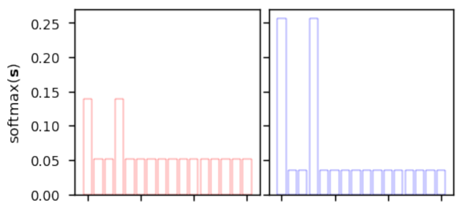
A Appendix - Supplemental Figures and Tables

Dev Metric			**EM**						**F1**			
Model N	1	2	4	8	16	32	1	2	4	8	16	32
albert-large-v2	0.793	0.867	0.904	0.932	0.960	0.983	0.825	0.890	0.919	0.942	0.966	0.987
albert-base-v2	0.790	0.874	0.916	0.945	0.967	0.984	0.818	0.894	0.929	0.954	0.972	0.987
albert-xxlarge-v1	0.860	0.909	0.933	0.950	0.962	0.984	0.889	0.927	0.942	0.955	0.965	0.985
roberta-base	0.753	0.842	0.900	0.944	0.973	0.987	0.787	0.866	0.916	0.954	0.980	0.992
distilbert-base-uncased	0.652	0.781	0.866	0.930	0.965	0.983	0.679	0.806	0.883	0.942	0.973	0.988

Table 3: Detailed top-N results for notable level-0 PCE models.



(a) s as F1 scores (red) and adjusted scores (blue)



(b) The two variants of s after applying $\mathrm{softmax}$

Figure 8: Example of target variants when a question is unanswerable; only the first and the fourth hypotheses are correct (no answer). The blue variant achieved a slightly better dev-set performance.

B Appendix - Manually Crafted Ensembles

We follow Occam's razor and attempt the following simple techniques for combining level-0 models.

1. **Plurality voting using the top-1 answers:** Each model m gives its top answer $a_m^{(i)}$ for question $q^{(i)}$; answers are normalized (using the same normalization method when computing scores [20]) then counted; finally, the normalized answer with the most votes wins.
2. **Weighted voting using probability scores of the top-1 answers:** similar to #1 except that each model's vote carries a weight equal to the probability score it gave its answer $P(a_m^{(i)})$. Even though the probability scores are not calibrated across models, they are still useful.
3. **Weighted voting using probability scores of top-1 answers and their statistics:** similar to #2 except that each top answer $a_m^{(i)}$ by model m for question $q^{(i)}$ is weighted by

$$\frac{P(a_m^{(i)}) - \mu(P(a_m^{(i)}))}{\sigma(P(a_m^{(i)}))}$$

   which measures how well m separates $a_m^{(i)}$ from the rest of answers it produced for $q^{(i)}$. We also experiment with $P(a_m^{(i)}) - \mathrm{median}(P(a_m^{(i)}))$ instead and report it below as #3′.
4. **Plurality voting using the top-N answers:** similar to #1 except that it considers the bag of all top-N answers produced by all models (order doesn't matter).
5. **Weighted voting using top-N rank as a penalty:** similar to the top-1 variant (#2) except that it penalizes lower-ranking answers using a linear scale; the top answer's weight is N, the second-best answer's weight is $N-1$, etc. — the N-th answer's weight is 1.
6. **Weighted voting using Fibonacci-scale of top-N rank as a penalty**: similar to #5 except that it uses the Fibonacci scale (using the n-th Fibonacci number instead of n).
7. **Weighted voting using probability scores of top-N answers:** a top-N variant of #2.
8. **Weighted voting using probability scores of top-N answers and their statistics:** top-N variants of #3 and #3′, reported respectively below as #8 and #8′.

The results of which are detailed below:

	#1	#2	#3	#3′	#4	#5	#6	#7	#8	#8′
Dev EM	83.119	84.090	83.580	84.090	18.657	31.902	78.217	84.288	83.876	**84.337**
Dev F1	85.463	86.134	85.726	86.149	32.588	42.538	80.618	86.256	85.947	**86.294**