Coattention, QANet, and Data Augmentation for Question Answering

Stanford CS224N Default Project

Wenqi Li
Department of Computer Science
Stanford University
wenqili@stanford.edu

Scott Xu
Department of Computer Science
Stanford University
scottxu@stanford.edu

Abstract

Our project aims to explore the effect of different Coattention techniques on the SQuAD question answering dataset, including the Dynamic Coattention Network [1] and the QANet [2]. Our Coattention network with dynamic decoder improved upon the baseline model by a small margin. Our final QANet implementation improved over the baseline model and achieved an F1 score of 69 on the dev set and 66.48 on the test set. We also employ several data augmentation techniques from the Easy Data Augmentation [3].

1 Key Information to include

- Mentor: Kathy Yu
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Our project tries to address the problem of Question Answering over the large SQuAD dataset, which contains over 100,000 questions on a set of Wikipedia articles [4], provided by Stanford University in 2016. Prior to the release of SQuAD, Question Answering datasets were either human annotated, high quality, but small in size, or semi-automated but unnatural in terms of type of reasoning required for question answering. While previous attention mechanisms on the encoder-decoder models achieved high improvement by learning to focus on the important part of word embeddings and hidden vectors, most of them are limited to the form of 1-D dot product with the embedding vector. The release of SQuAD gives researchers a chance to experiment deep neural networks with more innovative attention mechanisms, which we tried to implement in this project. Proposed by Xiong et. al. in 2017, the Dynamic Coattention Network [1] generalizes the techniques of word-level attention into larger contexts where questions, documents and other forms of input can share an coattention matrix to highlight their co-importance. In 2018, Yu et. al. proposed QANet, which extends on the coattention layer with convolutional model encoding layers, further boosting the performance of question answering on the SQuAD dataset.

3 Related Work

Many RNN and LSTM-based models have been applied to solve the question answering task on the SQuAD dataset since its release, including the BiDAF model [5] and the RNET [6]. To boost the performance, most transformer-based models start to apply various attention techniques inspired by the paper [7], using models full of convolutional layers and abandon the RNN and LSTM-based structures, such as the ones in DCN [1] and QANet [2]. However, all of these models are later beaten

by BERT-based models [8], where large, pretrained human language datasets like BERT are applied to provide useful embeddings to the aforementioned model structures. Our project doesn't apply any pretrained models and only aims to investigate the attention-based techniques that improve the BiDAF baseline.

In addition to getting large improvements over the SQuAD QA task, the method of coattention is quickly and widely adapted in later QA-related research, proving its generalizability to other contexts. For example, Liu et al. expanded on this method and implemented coattention layers to answer multiple-choice question [9], and Li et al. generalizes the approach to building coattention between document words and topical words for hashtag recommendation on microblogs [10]. The improvements in these applications proves the validity of coattention as a strong attention heuristic that is not strongly dependent on the type of the input corpus.

4 Approach

We experimented two approaches for the main model: the dynamic coattention model and the QANet model. Then, we implemented several data augmentations and ran our models on the augmented data.

4.1 The Dynamic Coattention Model

We implemented the Coattention model with dynamic decoder according to the Dynamic Coattentinon Network proposed in [1]. The model consists of two main parts: the Coattention-based encoder and the iterative dynamic decoder. The Coattention mechanism improves previous attention methods by proposing the concept of context-query attention in the QA task.

The dynamic coattention model uses an encoder-decoder structure in its design. In the encoding phases, we take the embedding of words in the questions, $(x_1^Q, x_2^Q, \cdots, x_n^Q)$, and the embedding of words in the documents, $(x_1^D, x_2^D, \cdots, x_m^D)$ and pass them to the same LSTM encoder to gets their hidden representations D and Q', defined as

$$d_t = \text{LSTM}_{\text{enc}}(d_{t-1}, x_t^D), \quad q'_t = \text{LSTM}_{\text{enc}}(q_{t-1}, x_t^Q)$$

for all $t = 1, 2, \cdots, n$, and we concatenate initially-zero sentinel vectors $d_\varnothing, q_\varnothing$ at the end of D and Q' to account for the case where the model doesn't attend any word in the respective input:

$$D = [d_1 \cdots d_m, d_\varnothing], \quad Q' = [q_1 \cdots q_n, q_\varnothing]$$

Finally, we feed the question representation Q' into a projection layer and a nonlinear layer to get the final representation for question $Q = \tanh(W^{(Q)}Q' + b^{(Q)})$. The rest of the encoding step uses a coattention encoder, aiming to produce the attention of both the question and document using their affinity matrix $L = D^\top Q$:

$$A^Q = \text{softmax}(L), A^D = \text{softmax}(L^\top)$$

Finally, we compute the matrix that includes both the attended context and attended question $C^D = [Q, DA^Q]A^D$ as the coattention context, and send it into a bi-directional LSTM to get the coattentional encoding U, where

$$u_t = \text{Bi-LSTM}_{\text{enc}}(u_{t-1}, u_{t+1}, [d_t; c_t^D])$$

For the decoder, we implemented the dynamic decoder with Highway Maxout Network (HMN). The LSTM decoder tries to predict the start point s_i and the end point e_i at each hidden state h_i. In particular, given the hidden state h_i, we compute for each word u_t its start score α_t and end score β_t by using two separate Highway Maxout Networks. A Highway Maxout Network takes in the current word vector u_t, the hidden state h_i, the previously predicted start and end points s_{i-1} and e_{i-1}, and compute a start (resp. end) score α_t (resp. β_t)

$$\alpha_t = \text{HMN}_{\text{start}}(u_t, h_i, s_{i-1}, e_{i-1})$$

Specifically, the HMN model has the following layers:

1. A non-linear $\tanh$ layer with learnable parameter $W^{(D)}$

$$r = \tanh(W^{(D)}[h_i; u_{s_{i-1}}; u_{e_{i-1}}])$$

2. The intermediate maxout layers with learnable parameter $W^{(1)}$ with bias $b^{(1)}$ and $W^{(2)}$ with bias $b^{(2)}$

$$m_t^{(1)} = \max(W^{(1)}[u_t; r] + b^{(1)}), \qquad m_t^{(2)} = \max(W^{(2)} m_t^{(1)} + b^{(2)})$$

3. the final maxout layer with learnable parameter $W^{(3)}$ with bias $b^{(3)}$

$$\text{HMN}(u_t, h_i, s_{i-1}, e_{i-1}) = \max(W^{(3)}[m_t^{(1)}; m_t^{(2)}] + b^{(3)}).$$

After computing the start and end scores, we make the predictions

$$s_i = \arg\max_t(\alpha_1, \cdots, \alpha_m), \qquad e_i = \arg\max_t(\beta_1, \cdots, \beta_m)$$

This iterative prediction ends when the pre-defined maximal number of iterations is reached. Finally, the predicted answer is the text span enclosed by the predicted start point and end point.

4.2 QANet

The QANet model follows the encoder-decoder structure and is made of five major components: the input embedding layer, the embedding encoder layer, context-query co-attention layer, the model encoder layer, and the output layer as described as below:

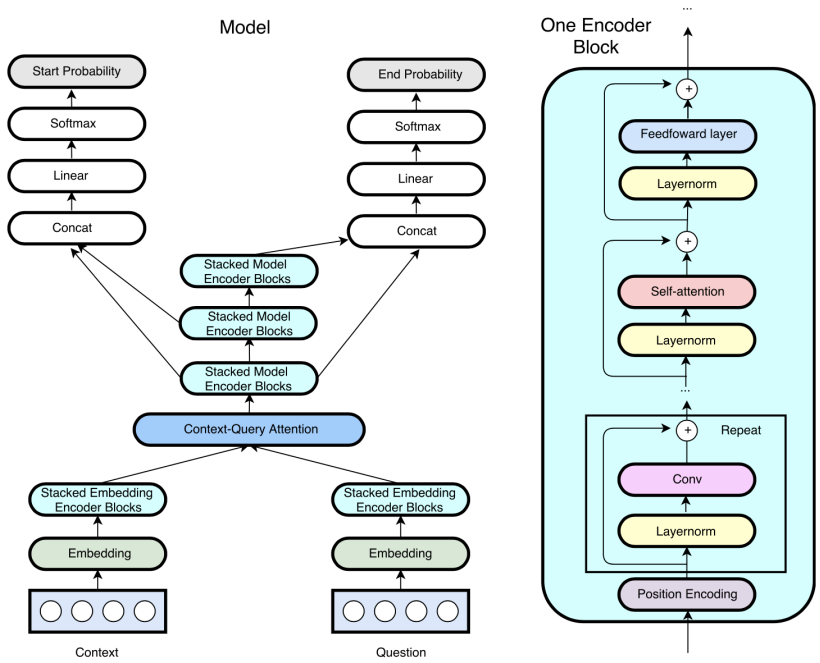


Figure 1: Model structure of QANet

First, the input embedding layer converts the word embeddings and character embeddings into an embedding layer. In our implementation, we feed character embedding into a 2D convolutional layer followed by a maxout layer to produce x_c in order to fit the dimension of the word embedding x_w, and concatenate them to be $[x_w, x_c]$. We take the concatenated embedding into a highway encoder as in the BiDAF model and apply another 1D convolutional layer to constrain its size to the hidden size of the model $d = 128$.

Next, the embedding encoder layer, shown on the right on the Figure 1, contains repeated convolutional layers wrapped by the layernorm layers. The position encoding layer is adopted by opensource code by BangLiu [1], which appends sinusoidal values on the embedding following the paper [7]. For the four convolutional layers used the repeated block in this module, we apply the depthwise

[1] https://github.com/BangLiu/QANet-PyTorch/blob/master/model/QANet.py

separable convolution layers with kernel size $k = 7$ following the design of the original paper. The resulting output is feed into a multihead self-attention module as in [7]. The final feedforward layers are two convolutional layers that use the model dimension d as the input and output dimensions.

After the above encoding blocks, we apply a modified coattention technique described in [1] that produces context-query attention scores. Denoting the encoded context as C and the encoded query as Q, the QANet model first computes the similarity score between each context-query pair, producing a similarity matrix $S \in \mathbf{R}^{n \times m}$ where n is the context length and m is the query length, using the similarity function

$$f(q, c) = W_0[q, c, q \odot c]$$

with learnable parameters W_0. Next, we apply softmax row-wise to get $\bar{S} = \text{softmax}(S)$ and compute the context-query attention as

$$A = \bar{S}Q^\top \in \mathbf{R}^{n \times d}$$

and compute the co-attention as

$$B = \bar{S}\bar{\bar{S}}^\top C^\top$$

where $\bar{\bar{S}}$ is given by applying column-wise softmax on S.

The following model encoding layer uses input $[c, a, c \odot a, c \odot b]$ where a, b are rows of the coattention matrices A and B. The input is then fed into the stacked model encoder blocks as in the second stage, and we use only 2 convolutional layers with kernel size of 5 on each of the blocks. We choose the number of blocks to be 5 due to memory constraints of the experiment.

Finally, the output layer takes the three matrices M_0, M_1, M_2 output by the model encoding layers and model the probabilities of start and end indices as

$$p_{\text{start}} = \text{softmax}(W_1[M_0; M_1]), \quad p_{\text{end}} = \text{softmax}(W_2[M_0; M_2])$$

with learnable weights W_1 and W_2.

4.3 Data Augmentation

To further facilitate learning, we attempted several methods to augment the training data. Since our computational resource allows only implementation-wise easy and cost-efficient methods, we use data augmentation methods inspired by the Easy Data Augmentation paper [3].

Specifically, two forms of data augmentation are implemented:

1. Random swap: for a given sentence in the context, randomly take two words in it and swap them.
2. Random deletion: Fix a probability p. For each word in a given sentence, randomly decide with probability p whether we delete it or not.

For each sentence in the context, we randomly (i.e. with probability 0.5) choose one of the above two operations to perform. Therefore in expectation, half of the sentences should have words swapped around, and the other half will have $100 * p$ percent of words deleted. In our implementation we chose $p = 0.1$.

Uniformly applying these two operations to all sentences to the context will result in changing the part that belongs to the correct answer span (e.g. answer gets deleted, or part of the answer gets swapped out of the answer span). Therefore, throughout we leave the parts in the answer span unchanged. We do not change the question in anyway in order not to destroy the semantic meaning of the question. The final result of this data augmentation is double the amount of training data, the extra being the data generated by the above procedure.

5 Experiments

5.1 Data

We use the SQuAD v2.0 dataset for question answering [4], and our experiments don't include other question answering datasets or pretrained language model parameters. We take advantage over word embedding and character embedding within the SQuAD dataset for training purpose.

5.2 Evaluation method

As described in the default project handout, our each experiment aims to maximize the F1 score of the prediction of starting and ending indices in the context. For the Coattention model, we uses the Cross Entropy Loss to evaluate the correctness of the model's output based on the output logits on the last layer of the model. For the QANet, we use the NLL loss over the output of the final log-softmax activation layer. These loss functions are equivalent and only slightly differs in implementation.

5.3 Experimental details

In the experiments, we train our model using the development set of SQuAD without data augmentation. For implementing the Coattention model, we mainly follow the design of the original paper and tune the hyperparameters of hidden dimension and embedding dimensions. We uses the Adadelta optimizer given in the baseline code with learning rate 0.001. The table below shows all the fixed hyperparameters in our experiments:

Parameter Name	Value
hidden dimension	100
embedding dimension	100
max epochs	50
decoding pooling size	16
decoding max iterations	4

For the QANet model, we apply various dropout layers after the convolutional layer, the self attention layer, and the feedforward layers in the encoder blocks using a self-increasing dropout rate of $\alpha l/L$, where l is the number of times we update the embedding by the layer output, denoted by the addition sign on the figure. We set $\alpha = 0.1$ as the default dropout rate and $L = b(n_c + 1)$, where n_c is the number of depthwise convolutional layers and b is the number of encoding blocks. We also attach simple dropout layers throughout the model using the default dropout rate.

For optimizer, we follow the design of the original paper by using an Adam optimizer with $\epsilon = 10^{-7}$, weight decay of 3×10^{-7}, and betas $(0.8, 0.999)$. We also use the same learning warmup techniques to exponentially increase the learning rate from 0 to 0.001.

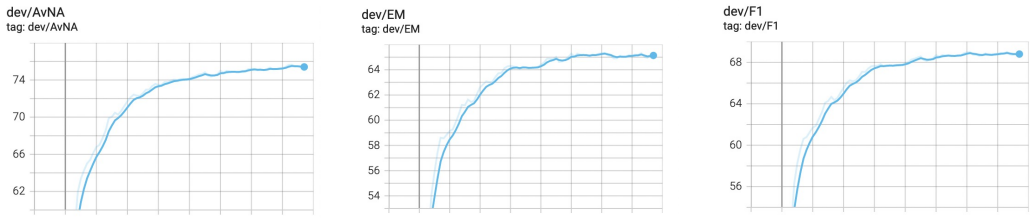
Additionally, we tune the following hyperparameter to get the best performance: number of model encoding layers, hidden size in the model, number of attention heads, and the batch size. We found that in general increasing the number of model encoding layer and the number of attention heads improves the performance, but a large combination of these two parameters may exceed the memory constraints in our machine. The only workaround is to decrease the batch size to 16, which approximately doubles the training time. In the end, the following configurations give the best result:

Parameter Name	Value	Suggested Value in [2]
hidden dimension	128	128
number of encoding layers	5	7
number of attention heads	8	8
batch size	32	32

5.4 Results

Since the coattention model has a similar structure to QANet and has worse performance over the non-augmented data, we decide to only experiment the data augmentation over the BiDAF baseline and the QANet model.

Parameter Name	AvNA	EM	F1
BiDAF baseline	67.55	57.49	60.77
BiDAF baseline (Augmentation)	68.07	57.80	61.34
Coattention	67.4	58.06	60.94
QANet	**75.63**	**65.32**	**69.00**
QANet (Augmentation)	73.77	63.97	67.46



(a) AvNA score (b) EM score (c) F1 score

Figure 2: Best QANet results on the dev set

Our best result on the test leaderboard is obtained by the QANet model without data augmentation, which has an EM score of **62.756** and an F1 score of **66.488**.

For the coattention model, we observed that in typical training, the F1/EM score would decrease at first as expected, then rapidly increase during a small amount of steps, and then stay horizontal with insignificant fluctuation in all steps after that. However, for some bad combinations of hyperparameters (e.g. a high learning rate or small hidden size), the evaluation loss is hardly decreasing, sometimes even increasing. The results is lower than expected and indicates that the current coattention mechanism has limitation in its expressive power and is not robust to change in hyperparameters like embedding dimension and hidden dimension.

For the QANet model, the results are as expected since it substantially increases over the baseline model. However, The performance of the model is sometimes random even when we use the same combination of hyperparameters, as the model may stuck in its local minima followed by an increase in the training loss. A possible solution to this is using a more dynamic scheduler in the training phase of the model to aggressively increase the learning rate when the training loss is rebouncing.

6 Analysis

We notice that the final QANet model is persistent in answering a phrase that matches the type of the question. In other words, when asked a "who" question, the QANet would answer a name of a group of people, and when asked a "when" question, it would answer a time, etc. This is the expected behavior in most situations, and it leads to good performance, but we notice that our model overdoes it. As an example, when the question asks "Who did Britain exploit in India?", the model answers "N/A" while the correct answer is "the Mughal state", which is a not a name or an individual of group of people, but human language shorthand for "the people that lives in the Mughal state". The model fails to realize that this is a phrase identifying a group of people, and thus predicted N/A.

As another example, when the question asks "the price of oil is usually a stable commodity until when?", the models answers a time period that appears in the context "1947 to 1967", but the true answer is "until the oil shock". We see that the model is unable to recognize the phrases "until the oil shock" as a time phrase, and leans towards to predicting an explicit time for a "when" question.

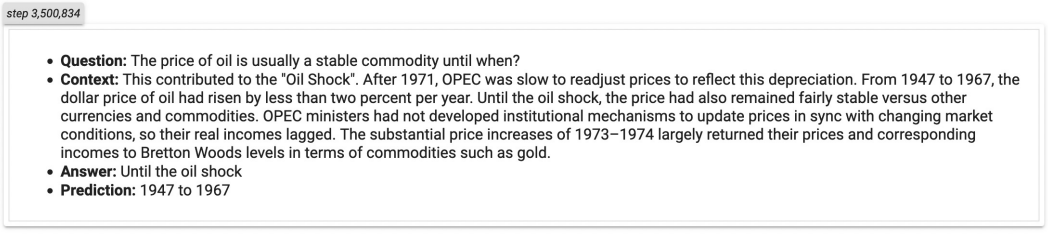


Figure 3: "When" Example

We believe that for the model to learn these subtleties in human language, and correctly recognize the property (who/when/what/how) of phrases, a more careful data augmentation would be helpful. Specifically, a sophisticated data augmentation method outlined in the original QANet paper [2] would be useful: translating the contexts into another language (e.g. French) and then back to English

to obtain a paraphrased version of the context. This can provide alternative wordings of the same meaning, and can help the model clarify potential implicit meanings of certain phrases, as discussed above. Another easier method outlined in [3] is to substitute words in the contexts by their synonyms. This provides alternative wordings as well, but does not change the structures of sentences, so the effect is supposed to be weaker.

We also observe that the model has an habit of copying the question. Namely, when the question asks "what theory ...", the model answers "thermodynamics theory", when the gold answer is "thermodynamics", when the question asks "which sea ...", the model answers "North Sea" while the gold answer is "North", when asked "what Republic ..." the model answers "Islamic Republic", while the gold answer is "Islamic". While the answers predicted by the model are reasonable, and sometimes even more natural (e.g. when a human is asked "what sea ...", they probably wouldn't just say "North"), this consistent behavior of repeating the question is interesting.

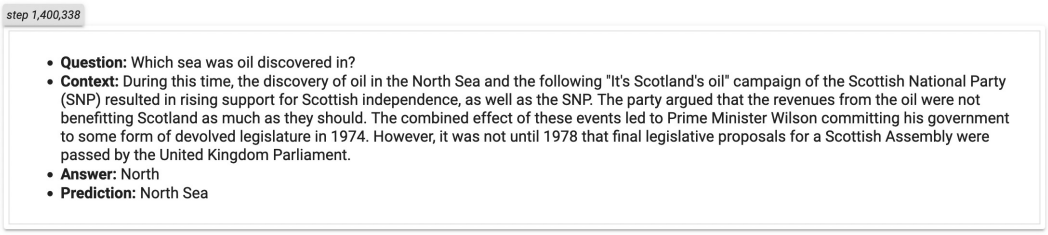


Figure 4: Question Repeating Example

We next discuss the effects of data augmentation. We notice that training on augmented data improves the baseline model but does not improve the QANet model. The reason may be two-fold:

1. The augmented data is not significantly different or better than the original data: only slight modifications are done, which give the model limited new information to learn from.
2. QANet does not contain any RNN structure, so the random swap augmentation method may has little effect on it. The performance could be possibly improved by ensembling QANet, RNet, and BiDAF, or using a larger model dimension and number of encoding blocks in each layers in the QANet model when more memory are available.

As we have mentioned above, translating the contexts back and forth would be a more effective data augmentation technique to employ, but our computational resources are limited, so we decided to choose the cost-efficient methods.

7 Conclusion

In this project, we implemented the Coattention dynamic network and the QANet model, both of which improved upon the baseline, with QANet model gaining the highest F1 score. Moreover, we applied data augmentation techniques, and the augmented training data improves the performance of the provided baseline model, but did not improve the performance of the QANet model. We recognize that our data augmentation techniques are limited by computational resources, but in doing this we also learnt the conclusion that extra training doesn't necessarily improve performance of an arbitrary model, and we are able to identify reasons why it does or does not help.

We hypothesize that several future directions may further improve our results and overcome the bottleneck of the current design of the QANet model:

1. Model ensembling. We found that a high percentage of recent advances on the SQuAD leaderboard use an ensemble of different models to predict the best output by a model voting mechanism or feature ensembling, which we haven't tried on either of our models. Considering that the attention-based model like DCN [1] and LSTM-based models like BiDAF [5] have significantly different designs but achieve similar results, it's reasonable to expect improvements by ensembling the QANet model with other types of models, possibly RNET [6].

2. Feature engineering. We found that the character embedding also boosts the performance of the model during our literature review, which suggests that more insightful representations could be fed into the model other than word and character embeddings. For example, it's possible to extract syntactic information like part of speech and grammatical structure of the input sentences and use them as the input of the proposed models.

References

[1] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*, 2017.

[2] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.

[3] Jason W. Wei and Kai Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196, 2019.

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

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[6] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Zhuang Liu, Kaiyu Huang, Degen Huang, Zhuang Liu, and Jun Zhao. Dual head-wise coattention network for machine comprehension with multiple-choice questions. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 1015–1024, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Yang Li, Ting Liu, Jingwen Hu, and Jing Jiang. Topical co-attention networks for hashtag recommendation on microblogs. *Neurocomputing*, 331:356–365, 2019.