CS230 Stock Price Prediction

Ryan Almodovar
Stanford University
ralmodov@stanford.edu

Vivek Misra
Stanford University
vmisra@stanford.edu

Abstract

Over the recent years, there have been several publications of deep learning models which integrate attention mechanisms that successfully extend to Machine Comprehension (MC). In order to understand how these architectures can improve rudimentary implementations, our task is to upgrade a baseline model by augmenting some of these recently introduced components to drastically increase prediction accuracy. On the Stanford Question Answering Dataset (SQuAD), our upgraded model improves the baseline F1 score from 43.93% to a new F1 test score of 71.68%.

1 Introduction

MC and Question Answering (QA) are both crucial tasks in natural language processing that require natural language understanding and world knowledge, as well as large-scale datasets to train learning models to accurately comprehend and predict correct answers from any given question. We explore various techniques to improve an existing model by implementing components from other successful recent models published, as well as tune the various hyperparameters in accordance to the changes made and feedback from the output results. One of the key components that adds the greatest improvement from the original baseline is implementing the Coattention Encoder layer originally introduced in the Dynamic Coattention Network (DCN) model (Xiong et al., 2017), as well as the Pointer Sentinel Mixture Model (Merity et al., 2016) to concatenate trainable sentinel vectors to the question and context word sequences. Both components and their integration into the existing project are described in detail in Section 3: Architecture and Approach.

2 Related Work

The model which our improvements are built upon are from the CS224N Winter 2018 SQuAD final project on GitHub[1]. The improvements are mainly based from the Coattention Encoder architecture introduced from the DCN model, which replaces the existing basic attention layer. The contextual embedding layers for the question and document that have been included in the base project are based on a simplified version of the Bidirectional Attention Flow (Seo et al., 2017) architecture, with the exclusion of the character CNN embeddings (Kim et al., 2016), and the substitution of the main attention layer with a basic attention layer. The recently released Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), provides over 100,000 question/document pairs and answers, allowing for a variety of qualities that culminate in a natural QA task. The performance based on the SQuAD dataset of the comprehension system is then uploaded and evaluated on CodaLab[2].

[1] https://github.com/abisee/cs224n-win18-squad

[2] http://codalab.org

3 Architecture and Approach

Figure 1: Architecture Overview

3.1 Contextual Embedding Layer

The GRU cells (Chung et al., 2014) in the existing model for the bidirectional RNN to form the encodings for question and document word embeddings are replaced with LSTM (Hochreiter et al., 1997) encoder word sequences to slightly increase the Dev EM and F1 scores from the baseline and Coattention models as seen in Table 1, and are defined as $X^Q = [x_1^Q, x_2^Q, ..., x_n^Q]$ and $X^D = [x_1^D, x_2^D, ..., x_m^D]$ respectively.

3.2 Coattention Layer

The next procedure after generating the contextual embeddings is to apply the attention layer between the two question and document embedding tensors. The following layer replaces the existing `BasicAttn` class from the base project with a new `CoAttn` class. The Coattention mechanism that attends to the question and document simultaneously is based on the Coattention encoder from the DCN model (Xiong et al., 2017), and fuses both attention question and document contexts. Figure 3 provides a visualization of the Coattention encoder.

3.2.1 Question and Document Encoder Sentinels

First, sentinel vectors $x_\varnothing^Q$ and $x_\varnothing^D$ (Merity et al., 2016) are concatenated to each embedding respectively, in order to allow the model to not attend to any particular word in the input as demonstrated by the illustration in Figure 2.

Figure 2: Visualization of the pointer sentinel-RNN mixture model. The query is used by the pointer network to identify likely matching words from the past. Probability mass can be directed to the RNN by increasing the value of the mixture gate g via the sentinel, seen in grey.

Let the resulting sequences be defined as the third-order tensors:

$$Q' = [x_1^Q, x_2^Q, ..., x_n^Q, x_\varnothing^Q] = [X^Q; x_\varnothing^Q] \in \mathbb{R}^{\beta \times \ell \times (n+1)} \tag{1}$$

$$D = [x_1^D, x_2^D, ..., x_m^D, x_\varnothing^D] = [X^D; x_\varnothing^D] \in \mathbb{R}^{\beta \times \ell \times (m+1)} \tag{2}$$

where β represents the dynamic batch size of the current iteration, and ℓ represents the hidden layer size.

Next, a non-linear projection layer is then applied on top of the question encoding to allow for variation between the question and the document encoding spaces, as defined by:

$$Q = \tanh(W_{ij}^{(Q)} Q'_{ajk} + b^{(Q)}) \in \mathbb{R}^{\beta \times \ell \times (n+1)} \tag{3}$$

where $W^{(Q)} \in \mathbb{R}^{\ell \times \ell}$ is a trainable weight matrix with Xavier initialization (Glorot et al., 2010) and $b^{(Q)} \in \mathbb{R}^{n+1}$ is a trainable bias vector with values initialized at zero. The Einstein summation convention is used to convey the product result of the matrix $W^{(Q)}$ and third-order tensor Q'.

Figure 3: Coattention encoder. The affinity matrix L is not shown here, but instead directly shows the normalized attention weights A^D and A^Q.

3.2.2 Coattention Encoder

To obtain affinity scores which correspond to all pairs of the question and document words, let the affinity tensor L be defined as:

$$L = D^T Q \in \mathbb{R}^{\beta \times (m+1) \times (n+1)} \tag{4}$$

Next, the affinity tensor is normalized with probability distribution with respect to the question dimension row-wise to obtain attention weights A^Q, and context dimension column-wise to obtain attention weights A^D via the softmax function:

$$\begin{align} A^Q &= \text{softmax}(L) \in \mathbb{R}^{\beta \times (m+1) \times (n+1)} \tag{5} \\ A^D &= \text{softmax}(L^T) \in \mathbb{R}^{\beta \times (n+1) \times (m+1)} \tag{6} \\ & \tag{7} \end{align}$$

The summaries, which are essentially attention contexts of each word of the question and the document respectively, are defined by:

$$\begin{align} C^Q &= DA^Q \in \mathbb{R}^{\beta \times \ell \times (n+1)} \tag{8} \\ C^D &= [Q; C^Q]A^D \in \mathbb{R}^{\beta \times 2\ell \times (m+1)} \tag{9} \\ & \tag{10} \end{align}$$

where the notation $[a; b]$ is defined as concatenation of tensors with respect to the ℓ dimension, and $C^Q A^D$ can be interpreted as the mapping of question encoding into space of document encodings.

Finally, a bidirectional LSTM fuses the temporal information to the Coattention context, and the sentinel vector $x^D_\varnothing$ is truncated, producing the attention output tensor U:

$$\begin{align} u_t &= \text{Bi-LSTM}(u_{t-1}, u_{t+1}, [d_t; c^D_t]) \in \mathbb{R}^{\beta \times 2\ell} \tag{11} \\ U &= [u_1, ..., u_m] \in \mathbb{R}^{\beta \times 2\ell \times m} \tag{12} \\ & \tag{13} \end{align}$$

where u_{t-1} and u_{t+1} are the forward and backward timesteps.

The resulting tensor U is then transposed so $U^T \in \mathbb{R}^{\beta \times m \times 2\ell}$ is compatible with the existing architecture to apply the matrix for masked context values $M^D \in \mathbb{R}^{\beta \times m}$, which is essentially a binary mask with a value of 1 when there is a real value, and a 0 for padding. Applying one last fully connected layer, which also reduces the vector space dimensionality of the ℓ dimension:

$$U' = \text{ReLU}(U^T) \in \mathbb{R}^{\beta \times m \times \ell} \tag{14}$$

which U' is then used as the input to the layer of computing the start and end probability distributions via a masked softmax function applied to the blended representation $[X^Q; U']$.

4 Experiments

4.1 Implementation and Metrics

The model is trained and evaluated on the SQuAD dataset, and uses pretrained GloVe word vectors on the Common Crawl corpus (Pennington et al., 2014). Running the experiments and viewing the results on TensorBoard displayed key metrics to determine optimal values for various hyperparameters. Each experiment was ran for approximately 10k to 15k iterations and generally stopped when the Dev F1 and EM scores continued to stay plateaued after several thousand iterations, indicating that the model was not improving despite the Training F1 and EM scores rising since by that time the model overfitting to the data was occurring.

(a) Dev F1 Score

(b) QAModel Loss

Figure 4: Dev F1 and QAModel Loss on TensorBoard after 12,000 iterations.

4.2 Hyperparameters

Based on the word length histograms of the Train and Dev datasets illustrated in Figure 5, the context word length from both sets were sparse past ∼450, so the hyperparameter `context_len` was reduced from 600 to 450 to optimize the run time for each training iteration. To further fit the model to allow higher complexity, the hyperparameter for the GloVe word embeddings `embedding_size` was increased from 100 to 200, which lead to more than a 10% F1 score increase. We use a dropout rate (Srivastava et al., 2014) of 0.2 to regularize our network during training to mitigate overfitting, and optimize the model using the ADAM Optimizer (Kingma et al., 2014).

(a) Context Word Length Frequency

(b) Question Word Length Frequency

Figure 5: Context and question word lengths of the provided Train and Dev datasets. Since the Train dataset is orders of magnitude larger than the Dev dataset, the frequency along the y-axis is normalized to visually compare the word lengths along the x-axis more accurately.

4.3 Results

The two evaluation metrics on the SQuAD dataset consists of exact match (EM) score, and F1 score. The EM score is evaluated based on the exact word sequence match between the predicted answer and a ground truth answer. The F1 score is evaluated based on the overlap between words in the predicted answer and a ground truth answer. Since there are some instances in which a document-question pair may have many ground truth answers, the EM and F1 scores for a document-question pair is evaluated as the maximum value across all ground truth answers. The Dev EM and F1 score results were recorded for each of the variations of the model by either running the project in `official_eval` mode or the result if the model was uploaded to the dev set on CodaLab, and are listed in Table 1. Finally, the overall metric is then averaged over all document-question pairs. The official SQuAD evaluation is hosted on CodaLab, which contains the training and development sets that are publicly available while the secret test set is withheld.

4.4 Analysis of Examples

Due to above limitations of our implementations and also dataset, we did observe certain adversarial examples along with good ones.Some of them are listed here: Example-1 " How" •

Paragraph:" quickbooks sponsored a " small business big game " contest , in which death wish coffee had a 30-second commercial aired free of charge courtesy of quickbooks . death wish coffee beat out nine other contenders from across the united states for the free advertisement ."

Question: how many other contestants did the company , that had their ad shown for free ,beat out?

True Answer: nine

Model predicts: nine

F1 Score : 1.000

EM Score: True

Example-2 " How " •

Paragraph:" the crew of apollo 8 sent the first live televised pictures of the earth and the moon back to earth , and read from the creation story in the book of genesis ,on christmas eve , 1968,which had been a troubled year for the us , marked by vietnam war protests , race riots , and the assassinations of civil rights leader martin luther king , jr. , and senator robert f. kennedy . " *Question*: how many other contestants did the company , that had their ad shown for free , beat out ?

True Answer:one-quarter

Model predicts: one-quarter

F1 Score: 1.000

EM Score: True

Example-3 " What" •

Paragraph:" in early 2012 , nfl commissioner roger goodell stated that the league planned to make the 50th super bowl " spectacular " and that it would be " an important game for us as a league ." *Question*: what one word did the nfl commissioner use to describe what super bowl 50 was intended to be ?

True Answer:spectacular

Model predicts: spectacular

F1 Score: 1.000

EM Score: True

Example-3 " What" •

Paragraph:" southern california is home to many major business districts . central business districts (cbd) include downtown los angeles , downtown san diego , downtown san bernardino , downtown bakersfield , southcoast metro and downtown riverside ." *Question*: what is the only district in the cbd to not have " downtown " in it 's name ?

True Answer: south coast metro

Model predicts: central business districts

F1 Score: 0.000

EM Score: False

So we see above model has limitations. This may be due to contextual neighbor support which is limitation of GloVe.

Example-4 " Which" •

Paragraph:“ prime numbers have influenced many artists and writers . the french composer olivier messiaen ”used prime to create unpredictable rhythms : the primes 41 , 43 , 47 and 53 .appear in the third e´tude ,“ neumes rythmiques ” .according to messiaen this way of composing was ” inspired by the movements of nature, movements of free and unequal durations ‘

Question: in which etude of neumes rythmiques do the primes 41 , 43 , 47 and 53 appear in ?

True Answer:the third e´tude

Model predicts: third

F1 Score: 0.667 *EM Score*: False This is same issue as in Example-3 Above observation suggests that if we have pre trained senetence level embedding it would probably solve contextual issues and accuracy can be improved.

5 Conclusion

We successfully implemented an end-to-end deep learning model for the ”CS 224N Default Final Project: Question Answering”. We were able to achieve promising preliminary results for this very challenging problem. Our accuracy and loss curve looks promising though further improvements are possible. As an outcome of this exercise, we wish to highlight certain improvements we wanted to make, certain issues which we faced and tuning which we did to achieve results.

5.1 Further Improvements

We definitely could have improved it by using character CNN embeddings but due to time constraints we did not fully implement it. With more time, we would love to investigate more hyper-parameter decisions. With more time, we would like to run a proper hyperparameter search algorithm over other parameters (e.g. batch size, LSTM hidden layer dimension, and hopefully converge on values that would boost our performance.

Other improvements that could have been completed but were not included are ensembling multiple models, and implementing the Highway Maxout Network (Xiong et al., 2017) as an intermediate layer between the Coattention layer and the output probability distribution layer.

We observed certain limitations of the SQuAD dataset which could have actually underplays the efficacy of our model. Since every answer to a question in SQuAD is a fixed pair of indices, the question answering task leaves no room for nuance and can mark other technically correct answers as incorrect.In short, our performance on the Question Answering task appears to be a good start, but our model certainly has limitations as described above along with the fact that there seems to be a theoretical limit on just how useful SQuAD can be as a proxy for measuring reading comprehension given the lack of nuance in answer choices.

5.2 Acknowledgments

We would like to thank Professor Richard Socher for teaching the various models and techniques present in the state-of-the-art publications of Natural Language Processing to allow for this project to be accessible, and the TAs of CS224N who consistently helped clarify core concepts from past assignments and questions from Piazza posts which were key to the implementation of this project.

References

[1] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic Coattention Networks for Question Answering *arXiv preprint arXiv:1611.01604*. 2016.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hananneh Hajishirzi. Bi-Directional Attention Flow for Machine Comprehension *arXiv preprint arXiv:1611.01603*. 2017.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation *In EMNLP, volume 14, pp. 1532–43*. 2014.

[4] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models *arXiv preprint arXiv:1609.07843*. 2016.

[5] Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. Character-Aware Neural Language Models *arXiv preprint arXiv:1508.06615*. 2015.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text *arXiv preprint arXiv:1606.05250*. 2016.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling *arXiv preprint arXiv:1412.3555*. 2014.

[8] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*. 1997.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *PMLR 9:249-256*. 2010.

[10] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way To prevent Neural Networks From Overfitting. *JMLR*. 2014.

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

Model	with variation	Dev/*Test* EM	Dev/*Test* F1
Baseline	*default*	**34.90**	**43.93**
	LSTM	**35.79**	**44.95**
Coattention Layer	*LSTM	49.87	64.99
	lower learning rate (0.0005)	49.08	63.50
	*higher dropout rate (0.2)	50.2	64.90
	*200D GloVe	***61.47***	***71.68***
References			
DCN (Xiong et al., 2017)	Ensemble	71.6	80.4
BiDAF (Seo et al., 2017)	Ensemble	73.3	81.1

Table 1: The variations marked with an asterisk* are included into the final model. The **bolded** values are the scores received after uploading to the CodaLab Dev set. The *italicized* values are the scores received after uploading to the CodaLab Test set.