Machine Question and Answering

Joseph Chang
Undeclared Undergraduate
Stanford University
chang100@stanford.edu

Malina Jiang
Department of Computer Science
Stanford University
malinaj@stanford.edu

Diana Le
Department of Computer Science
Stanford University
dianale@stanford.edu

Abstract

Machine comprehension, an unsolved problem in machine learning, enables a machine to process a passage and answer associated questions with a high level of accuracy that matches human performance. We use the SQuAD dataset, which simplifies the question-answer process, where the answer is defined by the starting and ending indices of its location in the context paragraph. In this paper, we explore several different models with a focus on the Multi-Perspective Context Matching (MPCM) model proposed by Wang et al [7]. The MPCM model features a relevancy matrix, used to filter out words in the context that are less relevant to the question, and uses a bi-directional LSTM encoding of the question and context. The model then applies attention mechanisms defined in the paper, including full-matching, maxpooling-matching, and meanpooling-matching to derive matching vectors, which are then decoded into starting and ending indices of the answer. In addition implementing and performing ablation on the features described in the MPCM paper, we also added an additional enforcement layer when determining the final indices of the answer, which conditions the ending index on the starting index. Our implementation of the machine comprehension model was able to achieve moderate results on the leader board, with an F1 of 57.45 and EM of 45.19.

1 Introduction

Reading comprehension involves reading and understanding a passage well enough to answer related questions correctly. With advancements in natural language processing and machine learning, models have been developed and improved to improve machine comprehension of written text. Despite these advancements, models at the current highest levels of performance, such as the Bi-Direction Attention Flow model proposed by Seo et al. (F1: 77.3, EM: 68.0) [4] and r-net ensemble model proposed by Microsoft Research Asia (F1: 84.0, EM: 76.9) [5], still lag far behind average human performance (F1: 91.2, EM: 82.3). These models also suffer from shortcomings not found in human readers, such as inability to answer questions of greater complexity or syntactical divergence between question and answer span within the paragraph. [6]

In previous models, the reading comprehension tasks were structured around and tested on children's books and the CNN/Daily Mail dataset, which used the bullet point summaries in news articles to test machine understanding by attempting to predict words removed from the summaries. Recently, the development of the Stanford Question Answering Dataset (SQuAD), based on more than 500 Wikipedia articles, has set a new bar for machine comprehension models. The SQuAD dataset

differs from its predecessors in that the answers can be found within the span of the paragraph, which greatly constrains the answer space and allow answers to vary from lengths of a single word to several sentences, as would be expected in real-world reading comprehension problems. Our model will explore and learn how to execute reading comprehension tasks on the SQuAD dataset by examining previous models on SQuAD data, with an emphasis on the Multi-Perspective Context Matching model proposed by Wang et al [7].

In the following sections, we will define the task in more detail, describe data and data preprocessing, as well as the architectural elements within our model. Finally, we will present the findings of our experiments with various models and the final results achieved by our model.

2 Task Definition

A reading comprehension task is comprised of a question q, answer a, and a context paragraph p that contains the span of the answer. Answering a question is defined as predicting the answer span within the context paragraph, where the starting and end indices (a_s and a_e) of the answer within the paragraph are determined by first finding the probability of each index being a starting or ending index in the answer and then taking a_s, a_e of the highest probability in the context, where the ending index is constrained to appear after the starting index. The answer accuracy over the dataset is then determined through two metrics, F1 score and exact match (EM) score.

3 Data

For training and validation purposes, we use the SQuAD dataset, which was divided into 81,386 training examples, and 4,284 validation examples (approximately 5% of the training set).

3.1 Data Preprocessing

The runtime of our algorithm is proportional to the output length of the paragraph. Additionally, from the histogram on the left which shows the lengths of the context paragraph, it is quite apparent that the majority of paragraphs are significantly less than the maximum paragraph length (766 words). In the interest of efficiency, we truncate the maximum output length to be 300.

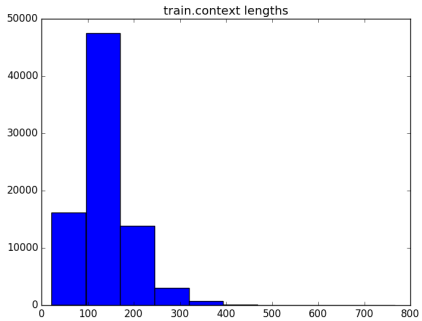


Figure 1: Histogram of Paragraph Length

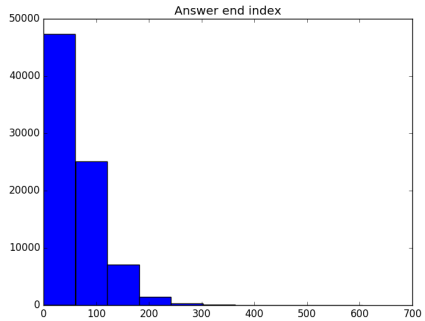


Figure 2: Histogram of end index

4 Architecture

4.1 Basic notation

For the sake of readability, we have defined the following variables:

- q the length of the question

- p the length of the paragraph
- $\mathbf{Q} \in \mathcal{R}^q$: the word embeddings of the question
- $\mathbf{P} \in \mathcal{R}^p$: the word embeddings of the question
- h: hidden state size of the LSTM
- d: GloVe vector embedding size
- l: Number of perspectives for our model

4.2 Relevancy Matrix

The purpose of this step is to filter out words that are irrelevant to the question before we encode the paragraph. We performed some elementary analysis on the lengths of the context paragraphs and the lengths of the answer and discovered that the average context paragraph in the training set was approximately 137.5 words long whereas the average length of the answer was only 3.4 words long. Because of this large discrepancy, even before encoding the question and answer, it is important to begin filtering out words that are irrelevant to the answer.

Consider the following example:

- **Question:** In what ocean are the Marshall Islands located ?
- **Answer:** Pacific Ocean
- **Paragraph (abridged):** The Marshall Islands , officially the Republic of the Marshall Islands (Marshallese : Aolepn Aorkin Maje) , [note 1] is an island country **located near the equator in the Pacific Ocean** , slightly west of the International Date Line . Geographically , the country is part of the larger island group of Micronesia . The country 's population of 53 ,158 people (at the 2011 Census) is spread out over 29 coral atolls , ...

As illustrated in this example, only a small portion (bold faced) is relevant to the question and the vast majority of the context paragraph is irrelevant. As such, we should begin filtering out words that have low similarity to the question. For each word in the paragraph, we define the relevancy score to be the maximum cosine similarity between the embeddings for the word from the paragraph and any word in the question. As such, we produce a relevancy vector $\mathbf{R} \in \mathcal{R}^p$. We then perform an elementwise multiplication between our relevancy vector and our paragraph word embeddings $\mathbf{P'} = \mathbf{R} \circ \mathbf{P}$. The end result of this step is that we weight words in the paragraph higher if they are similar to the words in the question.

4.3 Encoding

We utilize a bidirectional LSTM over the question ($\mathbf{Q}$) and paragraph ($\mathbf{P'}$) to generate a satisfactory encoding. A bidirectional implementation offers the advantage of using words appearing before and after the current word to generate the encoding of the center word whereas a unidirectional LSTM is limited by only using words before the current word to produce the encoding. According to the professors of this class, LSTMs are the "En vogue default model for most sequence labeling tasks" [1]. As such, it seems fitting and proper to utilize these to generate our encoding.

During the encoding process, because the answer is inherently highly dependent on the question, it is important to generate the paragraph embedding conditioned on the question. To do this, we first encode the question using an LSTM. Using these final hidden states over the question as the initial states for the paragraph encoding, we successfully create the conditionally encoding.

4.4 Multi-Perspective Context Matching Layer

This layer compares the contextual embedding of each passage with the accompanying question using three matching strategies as defined in the Wang et al paper. We first define dimensional weighted matchings with

$$m = f(v_1, v_2; W) \tag{1}$$

where v_1 and v_2 are two d-dimensional vectors, $W \in R^{lxd}$, l is the number of perspectives, and m is an l-dimensional vector $m = [m_1, ..., m_k..., m_l]$. Each element of m is a matching value from the k-th perspective and it is calculated using the cosine similarity:

$$m_k = cosine(W_k \circ v_1, W_k \circ v_2) \tag{2}$$

Here, W_k refers to the k-th row of W and $\circ$ refers to elementwise multiplication. Using these definitions, we can then define the three matching strategies with which to compare the contextual embedding of the passage with the question:

1. Full-Matching: the score is computed between the final state of question encoding and every word of the paragraph.

$$\overrightarrow{m_j^{full}} = f(\overrightarrow{h_j^p}, \overrightarrow{h_M^q}, W^1) \tag{3}$$

$$\overleftarrow{m_j^{full}} = f(\overleftarrow{h_j^p}, \overleftarrow{h_M^q}, W^2) \tag{4}$$

2. Maxpooling-Matching: For each word in the paragraph encoding, we retain the maximum score between this word and all words in the question encoding.

$$\overrightarrow{m_j^{max}} = \max_{i \in (1...M)} f(\overrightarrow{h_j^p}, \overrightarrow{h_i^q}; W^3) \tag{5}$$

$$\overleftarrow{m_j^{max}} = \max_{i \in (1...M)} f(\overleftarrow{h_j^p}, \overleftarrow{h_i^q}; W^4) \tag{6}$$

3. Meanpooling-Matching: For each word in the paragraph encoding, we retain the mean score between this word and all words in the question encoding.

$$\overrightarrow{m_j^{mean}} = \frac{1}{M} \sum_{i=1}^{M} (\overrightarrow{h_j^p}, \overrightarrow{h_i^q}; W^5) \tag{7}$$

$$\overleftarrow{m_j^{max}} = \frac{1}{M} \sum_{i=1}^{M} f(\overleftarrow{h_j^p}, \overleftarrow{h_i^q}; W^6) \tag{8}$$

The matching vector is then created by concatenating all of the forward and backward matches: $[\overrightarrow{m_j^{full}}; \overleftarrow{m_j^{full}}; \overrightarrow{m_j^{max}}; \overleftarrow{m_j^{max}}; \overrightarrow{m_j^{mean}}; \overleftarrow{m_j^{mean}}]$.

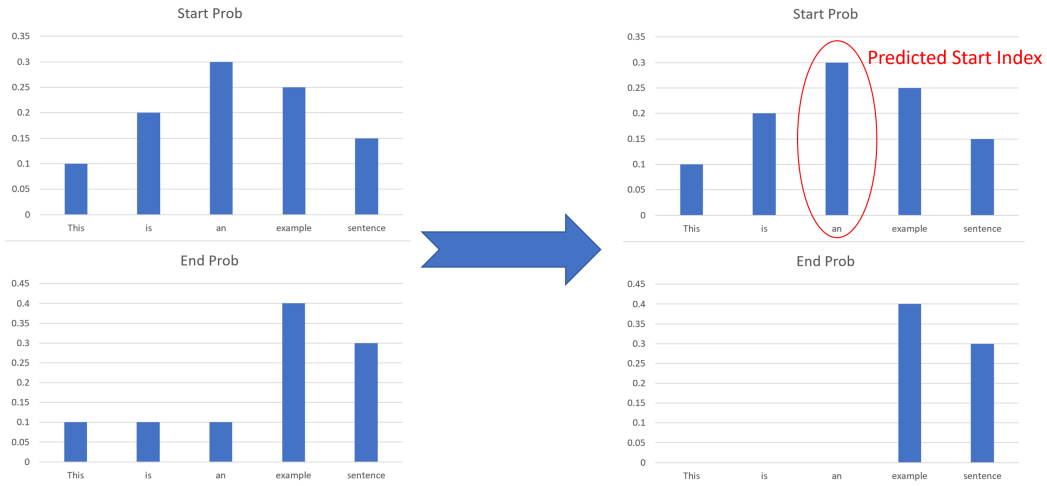
4.5 Decoder LSTM

This layer aggregates all of the matching vectors, so that each of the time steps of the passages can interact with the surrounding positions. The matching vectors are fed into a BiLSTM which generates the aggregation vector for each time step. After this, we utilize a feedforward neural network to generate probability distributions for the start and end indexes.

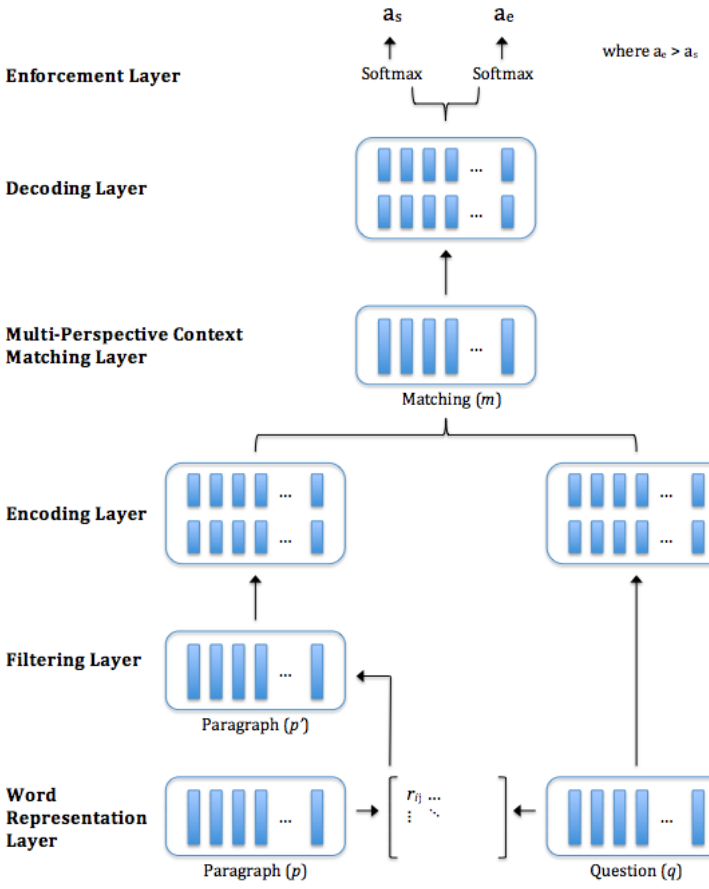
4.6 Enforcement and Prediction Layer

After the aggregation layer, we are given probability distributions for the start and end index. We find the predicted start index by taking the argmax of the start probability distribution. To enforce that the start index is before the end index, we modify the probability distribution of the end index so that for any index before the predicted start index, the probability is reduced to 0.

Additionally, when we predict the start and end index, we take the top 5 start and end probabilities. We compute the probability of the answer span by multiplying the the probabilty of the start index, end index, and the probability of the length.



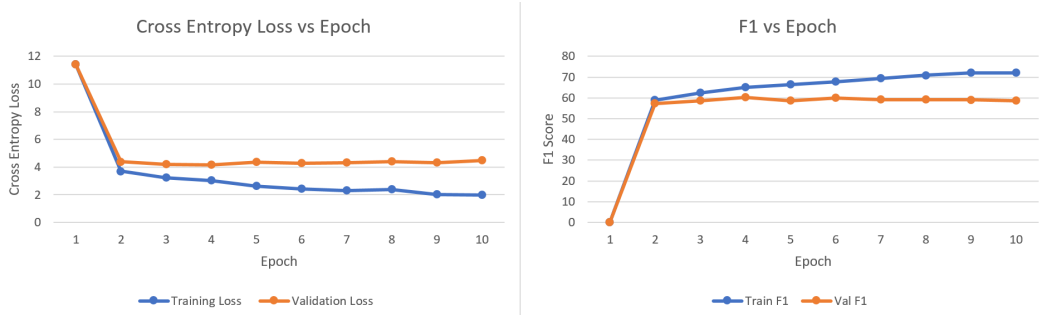
4.7 Diagram



5 Experiment Settings

We set the hidden size to 100 for all LSTM layers and set the number of perspectives to 10. To train, we minimize the softmax cross-entropy loss for the beginning and end points and an ADAM optimizer to update parameters. The learning rate was set to 0.01. The dataset's word embeddings

were also initialized with 300-dimensional GloVe word vectors from the 840B Common Crawl corpus (Pennington et al., 2014).



The above plots illustrate the cross entropy loss (left) and F1 score (right) vs epoch and clearly show overfitting.

To prevent overfitting, we employ dropout, which randomly drops units within our neural network during training. This is to prevent the units from co-adapting too much. Dropout also provides a way of combining exponentially many different neural network architectures efficiently by dropping certain units out, computing weights on the "thinned" networks, and then using a single neural network at test time without dropout. All of the units used at test time are scaled-down versions of the trained weights which are trained just enough to make some accurate predictions while not generalizing too much. [2]

After training on the training set, the matrices used in the Multi Perspective Context Matching layer ($\mathbf{W}^1, \cdots, \mathbf{W}^6$) overfit on the training set. In order to generalize better onto the validation, developtment, and test set, we implement L2 regularization on these matrices so that our model does not overfit as badly.

6 Results

With the full model, which involved all of the MPCM layers and dropout, the model was able to attain 60.04% F1 and 45.21% EM on the validation set. On the dev set of the leaderboard, the model was able to attain 57.30% F1 and 44.74% EM. On the test set of the leaderboard, the model was able to attain 57.45% F1 and 45.19% EM. The following sections involve experimentation with parameters and the structure of the model to observe changes in performance.

6.1 Layer Ablation

Model	F1	EM
without Full-Match	54.46%	39.89%
without Maxpooling	54.34%	39.45%
without Meanpooling	56.70%	43.02%
without Enforcement	55.67%	40.15%
Full	60.04%	45.21%

We evaluate the effectiveness of each of our layers. We built layer ablation models by removing one layer at a time. The table above shows the performance of the ablation models as well as our full MPCM model on the validation set after two epochs of training. Based on these results, we can see that removing any single component from the model significantly decreases the performance for both the F1 and EM score. Taking out the maxpooling-matching layer had the largest effect on the model's performance. Most importantly, we see that removing the enforcement layer, our creative addition, causes a significant drop in the model's performance.

6.2 Output Size Experimentation

Output Size	F1	EM
100	1.837%	0.4669%
300	57.31%	41.48%
500	53.39%	38.87%

We also experimented with changing the output size to find the optimal output size parameter for the model. Based off of the histograms generated in section 3.1, we had estimated that the optimal performance would occur around an output-size of 300-500, as this would capture the majority of paragraphs while also not wasting an computation on unusually long paragraphs. As shown by the table above, our highest performance occurred with an output size of 300. An output size of 100 drastically reduced the amount of information available to a model from the paragraph and may have even truncated an answer out of the paragraph. An output size of 500 also led to a slight decrease in performance; this may have been due to some over-fitting to longer paragraphs that did not allow for generalization.

6.3 Error Analysis

Analysis of our model's predicted answers and comparison with the answers provided by SQuAD showed that our model's prediction errors tended to fall under two common error types.

The first error was predicting a start index that appeared too early in context, which resulted in longer predicted answers than the actual answers provided by the dataset. As an example:

Question: What halftime performer previously headlined Super Bowl XLVIII ?
Predicted answer: Beyonc and Bruno Mars
Expected answer: Bruno Mars

In this case, the starting index has been moved left by 2 indices, resulting in extraneous words included into the final answer. This error is potentially caused by the enforcement layer of the model. Since enforcement aggressively ensures that the ending index will not appear before the starting index, it is likely that the model shifts the starting index to an earlier position in order to avoid cutting the predicted answers short of the actual end indices. Despite this possibility, we retained the enforcement layer due to the model improvements we generated. A possible resolution of this error to explore would be to penalize answers of longer lengths.

Another common error in our model is entity confusion, or the confusion of phrases of the same part-of-speech (e.g. noun phrases). As an example:

Question: Who did the Broncos prevent from going to the Super Bowl ?
Predicted answer: Dallas Cowboys , and Pittsburgh Steelers
Expected answer: New England Patriots

In this case, the model confuses different team names with each other and chooses the wrong team as an answer to the question. This error is possibly caused by the high similarity in word embeddings between different team names. One potential area to explore for this error is the addition of character embeddings, as they could potentially add more differentiation between similar phrases.

Though we implemented the full model detailed in the Wang et al. paper, our model did not achieve the same performance due to some changes in the parameters. Because of time constraints, we used an $l = 10$ instead of 50 in the original paper, and we did not use character embeddings. Both of these would have likely had a significant effect on our performance given that both provide additional information to the model.

7 Future & Related Work

We would like to continue implementing some of the ideas in the Multi-Perspective Context Matching paper. For instance, we would like to utilize character level embeddings as part of our word

representation as well as increase the number of perspectives from 10 to 50. Because of time constraints, we were not able to train character level embeddings and use high perspectives.

Additionally, we would like to use a more complex decoding process. Instead of predicting the start index and end index, we would attempt to label each word in the context paragraph as whether or not the word is part of the answer. After this, we utilize a viterbi decoder to label each word as (+1) if it is part of the answer and (-1) if it is not part of the answer. We then solve for the maximum subsequence sum and utilize this maximum subsequence as the answer.

8 Conclusion

In this paper, we build upon Wang et al.'s model of the Multi-Perspective Context Matching model by adding an additional enforcement layer that modifies the end index's probability distribution based on the predicted start index. Though we were not able to achieve the same success as the original paper, we find that our enforcement did increase the performance of the model on both the F1 and EM scores. Our experimental results also show competitive results on the CodaLab leaderboard.

References

[1] Richard Socher and Christopher Manning. 2017. Machine translation and advanced recurrent LSTMs and GRUs. http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture9.pdf

[2] Geoffrey Hinton. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In EMNLP, volume 14, pages 1532 43.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-Directional Attention Flow For Machine Comprehension. https://arxiv.org/pdf/1611.01603.pdf

[5] https://rajpurkar.github.io/SQuAD-explorer/

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. https://nlp.stanford.edu/pubs/rajpurkar2016squad.pdf

[7] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2017. Multi-Perspective Context MAtching for Machine Comprehension. https://arxiv.org/pdf/1612.04211.pdf