Unilateral Multi-Perspective Matching for Machine Comprehension

Sigberto Alarcon Viesca
Department of Computer Science
Stanford University
salarcon@stanford.edu

Jason Wang
Department of Statistics
Stanford University
zwang01@stanford.edu

Max Schorer
Department of Computer Science
Staford University
mschorer@stanford.edu

Abstract

Machine Comprehension (MC) involves answering a question with a given context paragraph. Recently, methods involving attention between the question and the paragraph have proved successful in this task. In this paper, we explore methods to address this challenge drawing inspiration from recent advances in the Stanford Question Answering Dataset (SQuAD) [1].

Our most successful model uses first encodes the question and the paragraph via a bidirectional LSTM. Next, we match the encoded sentences in two directions $P \rightarrow Q$ and $P \leftarrow Q$, drawing inspiration from the Bilateral Multi-Perspective Matching model [2]. This encodes the paragraph at each word against the encoded question from "multiple perspectives". We explore methods to aggregate these encodings to produce a knowledge representation. The knowledge representation is then fed into two bidirectional Long short-term memory (BiLSTM) decoders, one to predict the start index of the answer within the context and another for the end index. We discover that Unilateral Multi-Perspective Matching is a powerful tool and boosts our performances significantly, as demonstrated by our F1 and EM scores on the SQuAD Test Set of 66.2% and 54.6%, respectively.

1 Introduction

Machine Comprehension (MC) has long been a compelling challenge in Natural Language Processing, but until recently, it's been hampered by two limitations: modeling and datasets. On the first front, MC models were often comprised of multiple independent components that required a lot of custom feature engineering, resulting in long and complicated iteration cycles. However, the development of end-to-end neural network architectures has resolved these two issues, and these newer architectures are surpassing the previous state-of-the-art performance. Second, previous datasets had a trade-off in quality or size. High quality datasets had only a couple thousand observations, while didn't require much reasoning (ex: multiple chance, Cloze-style datasets where one word is randomly removed) and current models have already saturated performance. However, the recently developed Stanford Question Answer Dataset (SQuAD) addresses these issues by having a large corpus (100K observations) with questions whose answers require complex, human-like reasoning. This paper investigates the performance of a neural network architecture trained on this dataset.

2 Task Definition

A Machine Comprehension task involves a question, a paragraph containing the answer. The objective of the machine is to predict that correct answer span within the paragraph. In other words, we want to predict an answer span tuple $A = \{a_s, a_e\}$ given a question of length n, $Q = \{q_1, q_2, ..., q_n\}$ and a supporting context paragraph $P = \{p_1, p_2, ..., p_m\}$ of length m. Additionally, the answer must satisfy the constraint $1 \leq a_s \leq a_e \leq n$. The SQuAD task can be represented as learning probability $Pr(A|P, Q)$ from the training set and predicting answers by $A = \arg\max_{a_s, a_e} Pr(a_s|P, Q)Pr(a_e|P, Q)$ when generating answers given test data.

3 Dataset

The SQuAD dataset was built by first sampling 536 of the top 10,000 articles from English Wikipedia at random [1]. After selecting paragraphs from this corpus, crowdworkers from Amazon Mechanical Turk were hired to answer up to 5 questions on the content of those paragraphs given that the answer was a subsequence of the paragraph. Different types of reasoning are needed depending on the question (ex: lexical variation vs syntactic variation), while answers could come in a variety of forms, from a single date or number to a noun phrase (ex: property damage) or clause (”to avoid trivialization.”) A sample triplet taken from SQuAD is below, and the resulting dataset is a set of over 100,000 triplets (paragraph, question, answer) split into a training set (80%), development set (10%), and test set (10%). An example is shown below for reference:

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24201310 to earn their third Super Bowl title. The game was played on February 7, 2016, at Leviś Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the ”golden anniversary” with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as ”Super Bowl L”), so that the logo could prominently feature the Arabic numerals 50.

Question 1: Which NFL team represented the AFC at Super Bowl 50?
Answer 1: Denver Broncos
Question 2: Which NFL team represented the NFC at Super Bowl 50?
Answer 2: Carolina Panthers
Question 3: Where did Super Bowl 50 take place?
Answer 3: Santa Clara, California

Models trained on this dataset are evaluated on two metrics: F1 and Exact Match (EM). Humans scored 86.8% and 77% respectively, while the current highest score is already close to human performance (84%, 76.9%).

4 Models

In this section, we briefly explain several models we implemented and evaluated. We first describe our baseline model. Then we discuss our more successful model inspired by Bilateral Multi-Perspective Matching (BMPM) [2][3] as well as tweaks in attempts to improve its performance and to understand which architectures were most helpful. Each model consists of a combination of the following layers. We will refer to our best-performing model as Unilateral Multi-Perspective Matching as a reference to the methods of the BMPM

4.1 Layers

4.1.1 Word Representation Layer

In this layer, we represent each word in the question and the paragraph as a c-dimension vector where $c = 100$. The word embeddings are pre-trained and fixed during training. They are taken from GloVe (Pennington et al., 2014)[3]. The output is $P = [p_1, p_2, ..., p_m]$ and $Q = [q_1, q_2, ..., q_n]$ where p_i and q_j are c-dimensional word embeddings.

4.1.2 Context Representation Layer

Next, we process each word in the question (n words) and each word in the paragraph (m words). A bidirectional LSTM is used to process each word one at a time to generate a hidden states. The forward and backward hidden states for each word are concatenated and the outputs form the context representation $H = [h_1, h_2, ..., h_t]$ where $h_i \in \mathbb{R}^{2d}$ where d is the size of the LSTM hidden states.

4.1.3 Attention Flow Layers

The attention layers are responsible for linking and fusing information from the paragraph and the question. They are arguably important our models. Instead of popular attention methods that combine the question and the paragraph into a single vector, the attention layer outputs a knowledge representation where at each word in the paragraph is encoded with information from the question and vice versa. This is to ensure that we don't lose any information through summarizing.

We implemented various attention layers. The first was a basic attention layer that we benchmarked our results on. Next we created a Unilateral Multi-Perspective Context Matching Layer. In this implementation, we compare each contextual embedding of the paragraph with the question through multiple perspective. Finally, instead of encoding the paragraph through the context of the question, we reverse the order. We explore ways to encode each state of the question conditioned on the paragraph and use such information to add more information to the knowledge representation.

Baseline Simple Attention Layer

The is the key layer in the Baseline model. This layers computes each attention vector for each time-step from the paragraph with all time-steps in the question. We also call this step $P \rightarrow Q$ attention. The step determines which query words are most relevant to each paragraph word. We calculate a similarity matrix for each word in the paragraph p_i with each word in the question q_j. Then we take the softmax along each column and so the subsequent attended vector is a weighted average of the question context representation.

$$A = softmax(PQ^T)$$

$$Q_{attention} = AQ$$

Next we mix the context and attention vectors to produce the knowledge representation. First, we concatenate the paragraph and context-to-query attention vector. Next we multiply it by a matrix $W \in \mathbb{R}^{3d \times 3d}$ and add a bias. The output a knowledge rep $P_{knowledge} \in \mathbb{R}^{m \times 3d}$.

$$P_{knowledge} = [P; Q_{attention}; P \circ Q_{attention}]W + b$$

In the second method, we concatenate the context and attention vectors. The output is also a $\mathbb{R}^{m \times 3d}$ vector. In our experiments, the first method scored only slightly better than simple concatenation. Nonetheless, having either attention layer vastly improves performance.

$$P_{knowledge} = [P; Q_{attention}; P \circ Q_{attention}]$$

Unilateral Multi-Perspective Matching Layer

This is the key layer in the UMPM model. Here, we implement $P \rightarrow Q$ attention.

First, we compute the dimensional weighted matchings with

$$\boldsymbol{m} = f_m(v_1, v_2; W)$$

where v_1 and v_2 are d-dimensional vectors and $W \in \mathbb{R}^{l \times d}$ is a trainable variable. l is the number of perspectives. The output is a $l-$dimensional $\boldsymbol{m} = [m_1, m_2, ..., m_l]$. Each component is computed by

$$m_k = cosine(W_k \circ v_1, W_k \circ v_2)$$

and each m_k represents the value of the k^{th} perspective. W_k is the k^{th} row of W.

Next, we implement three matching strategies to compare one time-step of the paragraph with all time-steps of the question. Each is a perspective that tells us where important information exists in the paragraph. For sake of conciseness, we will describe how we compute each perspective for the forward states. The backward perspectives are also computed with similar methods.

1. **Full-Matching:** Each forward (or backward) contextual embedding h_i^p is compared with the last forward (or backward) embedding h_N^q of the question. We repeat this on the backward states.
$$\boldsymbol{m_i^{full}} = f_m(h_i^p, h_N^q, W_k)$$
2. **Maxpooling-Matching:** Each forward (or backward) contextual embedding in the paragraph h_i^p is compared with every forward (or backward) embedding h_j^q for the question. For each dimension, the maximum value is retained.
$$\boldsymbol{m_i^{max}} = \max_{j \in \{1,2,...,N\}} f_m(h_i^p, h_j^q, W_l)$$
3. **Attentive-Matching:** For each contextual embedding in the paragraph h_i^p and each context embedding in the question h_j^q, we compute a cosine similarity to represent weights.
$$a_{i,j} = cosine(h_i^p, h_j^q)$$
Then we weigh the question states h_j^q with the $a_{i,j}$ to generate a weighted sum of question embeddings at time-step i of the paragraph.
$$h_i^{mean} = \frac{\sum_{j=1}^{N} a_{i,j} h_j^q}{\sum_{j=1}^{N} a_{i,j}}$$

Finally, we match each mean vector with its corresponding attentive vector.

$$\boldsymbol{m_i^{att}} = f_m(h_i^p, h_i^{mean}, W_m)$$

At each time-step, we apply all three matching strategies for forward and backward states and concatenate the vectors to create a $6l$-dimension vector. We aggregate all vectors for the time-steps to create a knowledge rep.

$$P_{knowledge} \in \mathbb{R}^{n \times 6l}$$

$Q \rightarrow P$ Attention Layer

Our third and final model combines UMPM layer and a $Q \rightarrow P$ attention to explore if adding attention in the opposite direction improves performance. In this layer, we want to find which paragraph word have the closest similarity to one of the question words and thus critical to answering the question.

The following steps are inspired by BiDAF model [5]. First we compute the similarity matrix using the knowledge representations for the paragraph $P_{knowledge}$ and the question $Q_{knowledge} \in \mathbb{R}^{m \times 6l}$, which is the output of the UMPM layer where we reverse the context representation for P and Q.

$$S = softmax(P_{knowledge} \cdot Q_{knowledge}^T)$$

Next we obtain weights for each knowledge embedding in the paragraph by

$$b = softmax(\max_{col}(S)) \in \mathbb{R}^m$$

where the maximum along each column is retained. Let b_t be the weight for the t^{th} time-step in the paragraph. The weighted sum of each time-step

$$\tilde{p} = \sum_t b_t P^t_{knowledge}$$

is created and then tiled M times to create the matrix $\tilde{P} \in \mathbb{R}^{m \times 6l}$.

Finally, we concatenate the vectors to get the final knowledge representation.

$$P_{biAtten-knowledge} = [P_{knowledge}; P_{knowledge} \circ \tilde{P}] \in \mathbb{R}^{m \times 12l}$$

4.1.4 Aggregation and Prediction Layers

All our models use the same aggregation and prediction techniques. We feed the knowledge representation $P_{knowledge} \in \mathbb{R}^{m \times L}$ to a BiLSTM (concatenating the forward and backward outputs) to generate a layer of hidden states, $P_{answer\_s} \in \mathbb{R}^{m \times 2d}$ where d is the default size of the LSTM hidden state. A feedforward neural net is applied to each state of P_{answer_s} and a sigmoid is applied to the output to generate a probability distribution for $Pr(a_s|p, q)$.

P_{answer_s} is fed into a another BILSTM where P_{answer_e} is generated by aggregating the hidden states. A similar feedforward neural net followed by a sigmoid is applied to generate $Pr(a_e|p, q)$. We apply a second BiLSTM layer because such architecture ensures that a_e depends on a_s i.e. $a_e \geq a_s$.

4.2 Models

We concisely define our models as combinations of the layers described above. We provide an illustration of the UMPM in Figure 1.

4.2.1 Baseline

1. Word Representation Layer
2. Context Representation Layer
3. Baseline Simple Attention Layer
4. Aggregation and Prediction Layers

4.2.2 Unilateral Multi-Perspective Matching

1. Word Representation Layer
2. Context Representation Layer
3. Unilateral Multi-Perspective Matching Layer
4. Aggregation and Prediction Layers

4.2.3 Multi-Perspective + $Q \rightarrow P$ Attention

1. Word Representation Layer
2. Context Representation Layer
3. Unilateral Multi-Perspective Matching Layer
4. $Q \rightarrow P$ Attention Layer
5. Aggregation and Prediction Layers

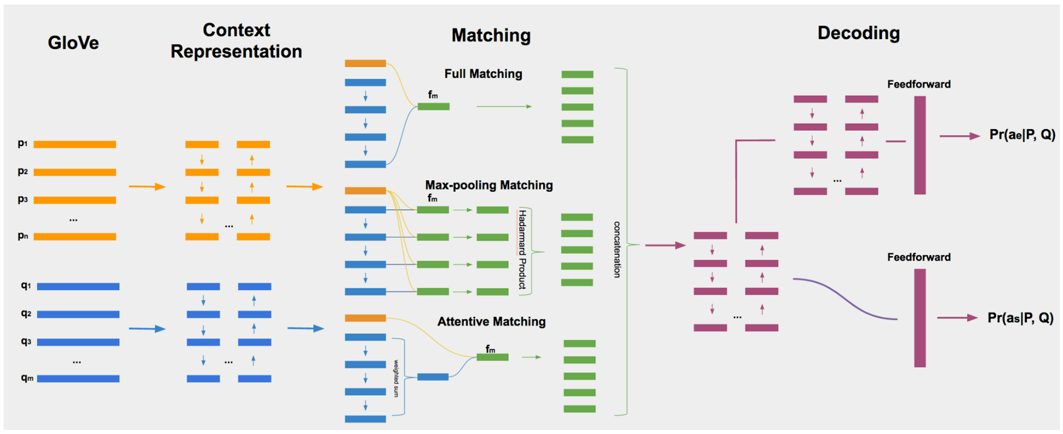

Figure 1: A graphic representation of our Unilateral Multi-Perspective Matching Model

Model	Train F1 / EM	Val F1 / EM	Dev F1 / EM	Test F1 / EM
Baseline	29.6 / 20.3	27.6 / 17.1	NA	NA
UMPM + $Q \rightarrow P$ Attention	65.6 / 48.2	56.1 / 42.0	NA	NA
UMPM	72.1 / 54.3	63.5 / 45.1	64.6 / 52.5	66.2/ 54.6

Table 1: Model performance as measured by F1 score and exact match (EM) on a percent scale. The Unilateral Multi-Perspective Matching (UMPM) model far outperformed the others in both the Training and Validation sets.

5 Experiments

5.1 Setup

We use 100-dimensional GloVe word embeddings as input, and truncate paragraphs and questions to maximum lengths of 300 and 25, respectively, for the sake of memory efficiency and speed. We applied masking to any states less than 300 or 25 that did not correspond to a word in the context/question. All our LSTM's have a hidden dimension size of 150 and we employ a dropout probability of 0.15 between each LSTM state to aid in generalization. We used an initial learning rate of 0.001 and annealed to half the previous value every time F1 score on the validation set remained relatively unchanged between two epochs. We trained each of our models for a minimum of 8 epochs, around half of which were at the initial learning rate. We use a number of perspectives (as defined for the BMPM model) of 30.

We initially constructed our vocabulary exclusively on the validation and training sets of SQuAD (cased). However, our first test on the development set showed a significant score in F1, from 63% in Val to 37%. Analysis on the output revealed a significant amount of <UNK> tokens, representing words outside the vocabulary we trained on. Therefore, we added lowercased and capitalized versions of all 6B GloVe vectors (uncased). With this vocabulary update alone, the same model achieved an F1 score of 64.6% on the Dev Set, demonstrating the value of a large vocabulary for generalization.

5.2 Results

The performance of our models as measured by F1 and exact match (EM) scores as defined by the authors of SQuAD is shown in Table 1. Unsurprisingly, the basic baseline model performed relatively poorly. The more sophisticated matching layer of UMPM achieved significantly better results. We attribute this to a more nuanced activation techniques, and the fact that the perspective matrix W allows for an added dimension through which two hidden states can interact. However, the UMPM + $Q \rightarrow P$ attention model performed worse than the UMPM. We hypothesize that attention

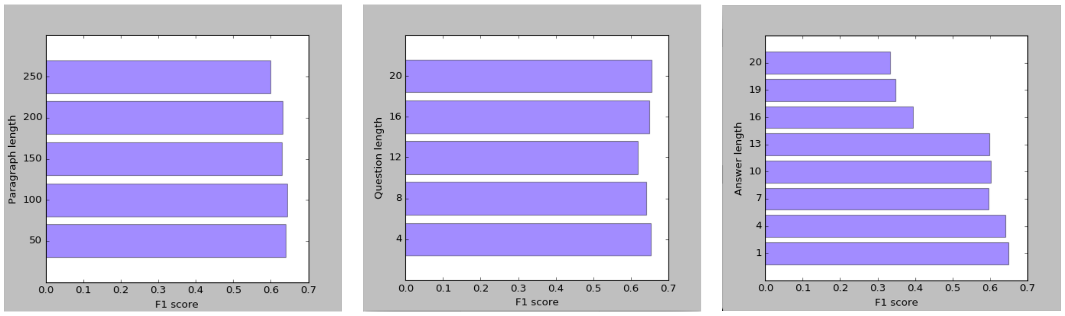

Figure 2: We compare the F1 score of our UMPM model against the length of the paragraph (left), question (center) and answer (right) in the validation set. While there is a negligible difference in performance when varying the context or question, performance is hindered as the answer length increases

was calculated on knowledge representations instead of the contextual representations. Applying attention twice may have weakened the effects of the first attention layer. A simple fix would be to look into would be to apply $Q \rightarrow P$ attention on the contextual representations instead.

Figure 2 shows how F1 score is affected by the lengths of the paragraph, question and answer. The biggest advantage of LSTM's and other more advanced recurrent networks is that they can preserve a ”memory” for multiple states, but they have their limits. We observe that the lengths of the paragraph and question had a negligible effect on F1. We interpret this as a successful representation of both, in th sense that the most important aspects for the task were maintained across states. However, we observer a very sizable drop in F1 as the answer span increases. We believe this could be because of three reasons. One, the BiLSTM step in the Aggregation layer loses too much information about the answer itself, leaving the decoding step (feedforward) with little information to make a prediction on. Two, the decoder architecture where the end index depends on the start index through a LSTM raises limitations for long answers where information is lost as LSTM pass through more stages. Three, $P \rightarrow Q$ attention is more effective where there are fewer words in the paragraph to attend to. Longer answers force us to pay attention to more words.

An observation of individual data points revealed that the easiest questions to answer are those with short answers that correspond to as specific fact/person/place, like ”where” ”when” and ”who”. These kinds of answers have a very specific format that makes them easier to learn. The fact that they are short also means that less information is lost between the answer start and answer end states. The questions where our models performed worse include more nuanced explanations whose syntactic and semantic format is more varied and longer. These are mostly ”Why” and ”How”. More often that the previous kinds of questions, these are open to interpretation.

6 Conclusion

In his paper, we present a series of models for the Machine Comprehension task; the most successful is Unilateral Multi-Perspective Matching, which combines elements from the Bilateral Multi-Perspective Matching model used for the Natural Language Sentence Matching (NSLM) task. Our best model achieved F1 and EM scores of 66.2% and 54.6% in the Test set, respectively.

The fact that question and paragraph lengths have negligible effect on performance makes us confident that we achieved a robust representation of them. However, we note that long answers suffer from a significant drop in both F1 and EM. In future work, we would experiment with more sophisticated aggregation and prediction layers to improve this. Instead of a single feed-forward network, we would attempt using deeper layers and also incorporate as input output of the UMPM layer Residual Networks demonstrated this is a powerful architecture in Computer Vision tasks [6].This may allow us to prevent lower performance in long answers as less information would be lost by the prediction step.

References

[1] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

[2] Wang, Z., Hamza, W., & Florian, R. (2017). Bilateral Multi-Perspective Matching for Natural Language Sentences. arXiv preprint arXiv:1702.03814.

[3] Wang, Z., Hamza, W., Mi, H., Florian, R. (2016). Multi-Perspective Context Matching for Machine Comprehension. arXiv preprint arXiv:1612.04211

[4] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).

[5] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional Attention Flow for Machine Comprehension. arXiv preprint arXiv:1611.01603.

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).