Question Answering on the SQuAD Dataset Using Multi-Perspective Context Matching

CS224n final project

March 21, 2017

Stanislav Fort, sfort1@stanford.edu
Samual Colbran, samuco@stanford.edu
Stanford University

Abstract: In this paper we present an implementation of the multi-perspective context matching neural architecture for question answering on the Stanford Question Answering Dataset (SQuAD). We performed training on a set of 81,400 examples and achieved an F1 score of 47.516 % and EM of 34.249 % on an unseen test set with a compact model of only 173,882 trainable parameters. We used 100 dimensional GloVe word vector embeddings as the input to our model. We describe the model in detail, present our training strategy and results, and discuss hyperparameter selection. An analysis of probability distributions over a few example answers is also presented to further explore the model's behavior. A simple comparator baseline is also implemented and studied.

Contents

1 Introduction

Machine comprehension is a critical part of a large number of Natural Language Processing applications such as virtual assistants, chatbots, and information retrieval systems. Until recently, there has not been an accessible dataset to evaluate models on. [1] The development of the Stanford Question Answering Dataset (SQuAD) enables researches to develop and test realistic models of machine comprehension.

In this paper, we implement a modification of the multi-perspective context matching architecture for machine comprehension that was first presented in [1]. We also implement a simple a baseline model for verification. Our multi-perspective architecture uses pretrained GloVe vectors, bidirectional LSTMs and a very important matching layer that tries to understand the relationship between the given context paragraph, question, and answer. For each word in the context passage, its representation is matched against the question representation from a number of perspectives, which are then used in parallel to make an answer prediction.

2 Datasets

2.1 The Stanford Question Answering Dataset (SQuAD)

The SQuAD contains 100K question-answer pairs (along with a context paragraph). The answer is contained within a span in the context paragraph (i.e. it does not rely on prior knowledge outside of the context and does not contain tokens split up over the context). A histogram of the context paragraph lengths is shown in Figure 2. For questions, the histogram is shown in Figure 1. The following is an example question-answer-context pair from the SQuAD

> **Question:** Why was Tesla returned to Gospic?
> **Context:** On 24 March 1879, Tesla was returned to Gospic under police guard for **not having a residence permit**. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
> **Answer:** not having a residence permit (from the context span 14 - 18 that is bolded above)

An important aspect required to model the dataset is the length of context and questions. While we would ideally like to create a model that could handle questions and context paragraphs of arbitary length, the models described in later sections impose a set sequence length and enforce this using padding or clipping. To reduce memory requirements and calculation size, we desired a small sequence length that was still able to handle the majority of questions. To better visualise the dataset we created histograms of the length of the sequences, as shown in figures x and y. This let us pick sequence lengths that ensured the model was fast and memory efficient while losing (minimal) accuracy. We did not have access to the test dataset and assumed that it followed a similar distribution.

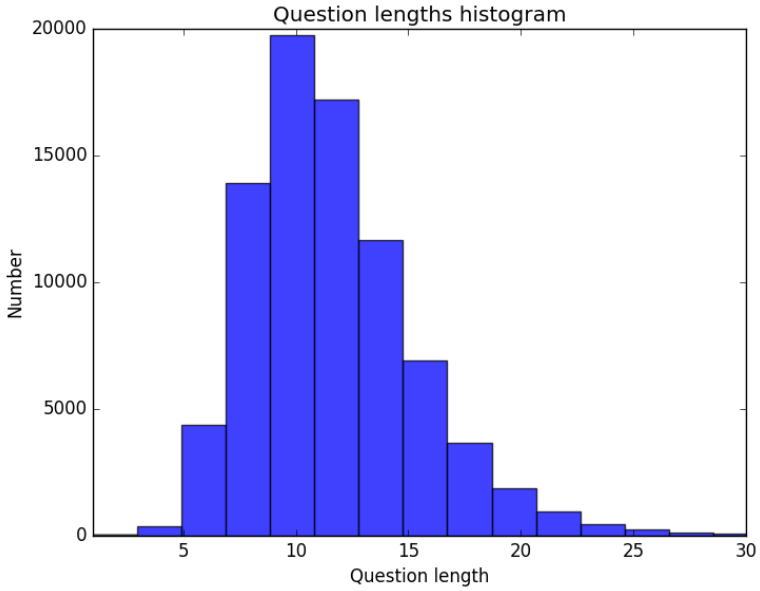

Figure 1: The histogram of question lengths in the training set.

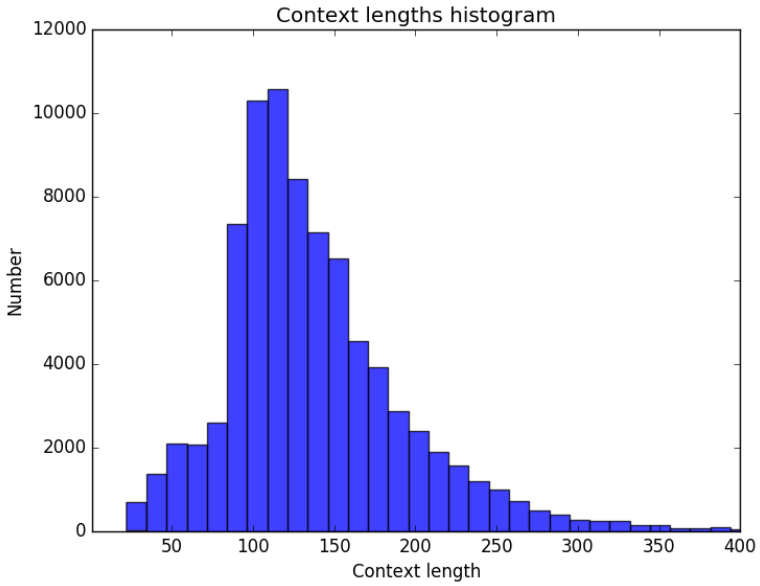

Figure 2: The histogram of context paragraph lengths in the training set.

2.2 Global Vectors for Word Representation (GloVe)

We use 100 dimensional word vectors trained on the 6GB word corpora (Wikipedia + Gigaword) using the GloVe algorithm. Rather than representing individual words in the vocabulary, we represent words as vectors of smaller dimensionality that capture meaning. Not only does this speed up huge matrix multiplications, the word vectors capture and model relationships between words. One challenge with the pre trained word vectors is that many words in the SQuAD dataset do not appear in the vocabulary, which results in many UNK tokens. To work around this, we predict a span and then extract the original tokens from the context (even if our model doesn't understand them). This means that if it predicts an UNK, the correct

corresponding token will be inserted in the answer. We are using a vocabulary of 115,613 words, which produces the unknown word token very rarely on our dataset.

3 Baseline model

For initial development and validation, we implement a simple baseline model with attention. The schematic of the model is shown below. We do not go into further detail as this model was only a baseline. A very thorough description of our advanced model can be found in Section 4.

1. Question and context embeddings are Q and C are loaded.
2. Question is passed through a bidirectional LSTM and Q' generated.
3. Context is passed through a different bidirectional LSTM and C' generated.
4. Attention matrix A is calculated for each pair of a context word and a question word as a dot product between their hidden states and then softmaxed.
5. A modified version of the context C' is calculated as AQ' and concatenated to the original context C'.
6. A simple softmax with one neuron classification is done on the concatenated matrix to obtain the probability that a particular context word is a start of the answer,
7. The same is done with a separate classifier for the end of the answer.

4 Multi-perspective context matching model

The main model we used in this project was a modified version of the multi-perspective context matching algorithm presented in [1]. A diagram of the model's architecture taken from [1] is shown in Figure 3.

4.1 Naming convention

We implemented a modified version of the multi-perspective context matching algorithm presented in [1]. In this section, we present the detailed inner mathematics of the model. We will be explicitly stating which indices are summed over rather than using matrix and vector notation as in our opinion it is clearer.

Let the context C be a matrix $\{C_{ace}\}$, where a is the index of a particular example in a batch, c the index of a particular word in the context paragraph, and e the index of the particular component of the word embeddings representation (in our case $GloVe$). C_{ace} is then the e^{th} component of the word embedding of the c^{th} word of the context of example a. Let Q be an equivalently defined matrix for the question, such that $\{Q_{aqe}\}$ are its elements.

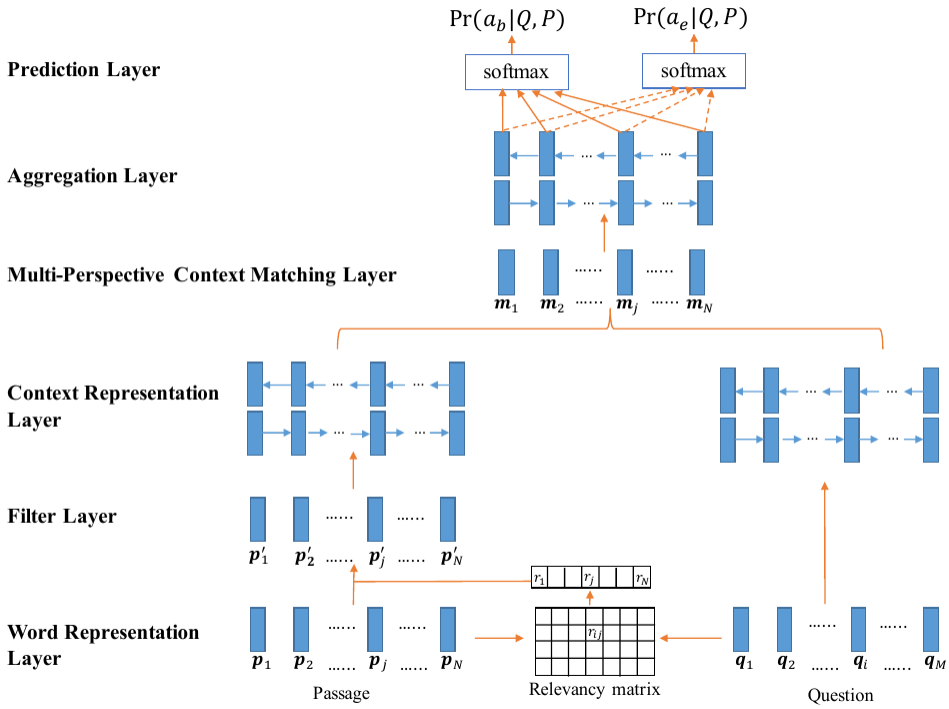

Figure 3: A diagram of the multi-perspective context matching algorithm for text comprehension implemented in this project. The figure comes from the original paper [1].

4.2 Filter layer

The context as a whole contains a lot of information. A question typically asks only for a specific small section of the context. To highlight the relevant parts of the context, we use a filter layer, as done in [1].

The filter matrix R_{qc} is defined as

$$R_{aqc} = \sum_e Q_{aqe} C_{ace} \,, \tag{1}$$

i.e. for a particular example a, the filter matrix for a particular context word c and question word q is the dot product of the two word embeddings. As the word embeddings contain semantic and syntactic information about the words, a high dot product signifies high relatedness of the words. Even more directly, the filter matrix will be high if the word in the context c is the same as the question word q. This helps identifying the relevant parts of the context, and works as **attention**.

To simplify the resulting object, let the filter vector r_{ac} be

$$r_{ac} = \max_q R_{aqc} \,, \tag{2}$$

i.e. the filter vector for a particular context word c is the maximum dot product of this word with every word in the question. For example, if a similar word appears in the question, the word is highlighted in the context. To apply this in the context, we perform

$$C'_{ace} = r_{ac} C_{ace} \,.$$

The context words therefore get highlighted by the amount of similarity to the most similar word in the question.

4.3 Initial encoding using independent LSTMs

As the next step, we applied a bidirectional LSTM on the *filtered* context representation C'_{ace}, along the c dimension. The LSTM moves along the words of the context and at each word generates an output vector of length of our hidden state length. Effectively, the C'_{ace} is therefore mapped by the forward run of the LSTM to another matrix $\tilde{C}^{\text{fw}}_{ach}$, where the last index h now indexes along the newly generated hidden state by the LSTM. We applied the LSTM bidirectionally, and therefore also got the backwards pass result $\tilde{C}^{\text{bw}}_{ach}$. Let the final hidden state of the forward pass of the LSTM be $\tilde{c}^{\text{fw}}_{ah}$ and $\tilde{c}^{\text{bw}}_{ah}$ for the backwards pass.

A separate bidirectional LSTM with new parameters was then applied to the original question representation (no filtration applied to the question) $\{Q_{aqe}\}$, equivalently generating $\tilde{Q}^{\text{fw}}_{aqh}$, $\tilde{Q}^{\text{bw}}_{aqh}$, and $\tilde{q}^{\text{fw}}_{ah}$, $\tilde{q}^{\text{bw}}_{ah}$.

These new representations of the word vectors generated by the LSTMs are better suited for our task as they are aware and influenced by the surrounding words in the context and question respectively. They are more custom-fitted than the generically generated word embeddings such as GloVe - instead, they are the specific context and question sensitive word embeddings.

4.4 Matching

The crucial part of the model is the matching layer. In the matching layer, dot products of the results of the LSTMs on the question and filtered context representations are generated, with an a important middle step of applying a **perspective**.

A perspective P_h is a vector of hidden state index h. To apply a perspective P to a particular word embedding (might have been through LSTMs or other transformations first), one performs

$$V^P_h = V_h P_h \,. \tag{3}$$

In other words, a vector is point-wise multiplied by a perspective vector. Perspective vectors are in our model represented as rows of a matrix W. 6 different matching representations are then generated by applying perspective vectors and looking at dot products on different input vectors. They are described in detail as follows.

4.4.1 Full matching

$\tilde{C}^{\text{fw}}_{ach}$ and $\tilde{C}^{\text{bw}}_{ach}$ are used. We will only describe the procedure on one of them, as the other is equivalent. These are matched with the last hidden state of the forward pass on the question $\tilde{q}^{\text{fw}}_{ah}$, or its backward pass $\tilde{q}^{\text{bw}}_{ah}$. Let the matrix of projections be W. First an auxiliary matrix is calculated as

$$U_{akch} = W_{kh}\tilde{C}^{\text{fw}}_{ach} \,, \tag{4}$$

where k is the k-th projections vector in the projection matrix W. The vector U is the projection of the context word c onto the projection k. The second auxiliary matrix is defined as

$$V_{akh} = W_{kh}\tilde{q}^{\text{fw}}_{ah} \,, \tag{5}$$

which is the projection of the last hidden state of the question LSTM forwards pass onto the projection direction k. Then, the result of the *full matching* is obtained as

$$M_{akc} = \sum_h U_{akch} V_{akh} \,. \tag{6}$$

The result can be intuitively understood as the dot product between each context word c projected into the projection k with the last hidden state of the LSTM applied to the question projected into the same projection k.

Two matching matrices are generated. $M^{\text{full,fw}}_{akc}$ for the forward passes using $\tilde{C}^{\text{fw}}_{ach}$, $\tilde{q}^{\text{fw}}_{ah}$ and projection matrix W_1, and $M^{\text{full,bw}}_{akc}$ for the backwards passes using $\tilde{C}^{\text{bw}}_{ach}$, $\tilde{q}^{\text{bw}}_{ah}$ and projection matrix W_2.

This matching is sensitive for questions where the last hidden state on either side is sufficient for answering.

4.4.2 Max-pooling matching

Here, the auxiliary matrix V is calculated from the LSTM outputs along the question $\tilde{Q}^{\text{fw}}_{aqh}$ and $\tilde{Q}^{\text{bw}}_{aqh}$. It is calculated as

$$V_{akqh} = W_{kh} \tilde{Q}^{\text{fw}}_{aqh} \,, \tag{7}$$

for the forwards pass and equivalently for the backwards one. The matching is then calculated as

$$M_{akc} = \max_q \left(\sum_h U_{akch} V_{akqh} \right) . \tag{8}$$

It therefore gives the highest dot product matching between k projection of the context word c and the k projection of the question word q over all question words. This matching allows us to use information from the middle of the question as well.

Two matching matrices are generated again – $M^{\text{maxp,fw}}_{akc}$ with W_3 and $M^{\text{maxp,bw}}_{akc}$ with W_4.

4.4.3 Mean-pooling matching

The mean-pooling works the same as max-pooling, with the only difference coming at how the matching is calculated from the auxiliary matrices. Here,

$$M_{akc} = \text{mean}_q \left(\sum_h U_{akch} V_{akqh} \right) . \tag{9}$$

The average matching over all question words is then obtained. This gives a matching that is not focused on a particular question word, but rather looks at the average over all question words.

Two matching matrices are generated – $M^{\text{meanp,fw}}_{akc}$ with W_5 and $M^{\text{meanp,bw}}_{akc}$ with W_6.

4.5 Aggregation

All six perspective matchings use separate matrices of perspectives $W_1, ..., W_6$. Each matrix contains l projection vectors, that are all separately trainable. The big advantage of the model lies in the ability to choose the important directions in the dataset using this approach.

The resulting matchings are concatenated along the k direction into a large matching vector

$$M_{alc} = \left[M^{\text{full,fw}}_{a\_c}, M^{\text{full,bw}}_{a\_c}, M^{\text{maxp,fw}}_{a\_c}, M^{\text{maxp,bw}}_{a\_c}, M^{\text{meanp,fw}}_{a\_c}, M^{\text{meanp,bw}}_{a\_c}\right] \tag{10}$$

where $\_$ is the index over which the matrices are concatenated. The matrix now has dimension $[\text{batch}, 6 \times \text{perspectives}, \text{context}]$.

4.6 Final LSTM on the matchings

The matching matrix M_{alc} aggregates information about the relationship between the context and question at different "directions" of the embedding space and generated using different summing method as described above. To let the different matchings interact, we apply a bidirectional LSTM on M_{alc} along the c (context words) direction. At each step, a state of size $6 \times$ perspectives is transformed into a new vector. The outputs of the LSTM are A^{fw}_{ach} and A^{bw}_{ach} for the forward and backward passes respectively. We then concatenate them along the hidden direction h to a knowledge matrix

$$K_{ach} = \left[A^{\text{fw}}_{ac\_}, A^{\text{bw}}_{ac\_}\right] \ . \tag{11}$$

This matrix now represent our complete relevant information about the context word c.

4.7 Classification on knowledge K

The last is applying a single linear neuron to the knowledge K_{ach} to get a real number for each position in the context c representing our belief that the position is the start of the answer. The same classification is done with different matrices for the end of the answer. The assumption here is that the start and end predictions are *independent* at the last step, and that all dependence has been captured in the knowledge K.

The predictions are

$$\text{Start}_{ac} = \sum_h K_{ach} W^{\text{start}}_{hc} + b^{\text{start}}_c \,, \tag{12}$$

$$\text{Stop}_{ac} = \sum_h K_{ach} W^{\text{stop}}_{hc} + b^{\text{stop}}_c \,, \tag{13}$$

for matrices W^{start}_{hc} and W^{stop}_{hc} and bias vectors b^{start}_c and b^{stop}_c. The *softmax* is applied to both vectors to get probabilities out.

4.8 Loss

We are using the cross-entropy loss. Let the correct one-hot start label be $\hat{y}^{\mathring{start}}_{ac}$, and the correct one-hot end label be $\hat{y}^{\mathring{stop}}_{ac}$. Let the softmax predictions be $y^{\mathring{start}}_{ac}$ and $y^{\mathring{stop}}_{ac}$. Then the cross-entropy loss per batch is

$$\text{CE} = -\text{mean}_a \left(\sum_c \hat{y}^{\mathring{start}}_{ac} y^{\mathring{start}}_{ac} + \hat{y}^{\mathring{stop}}_{ac} y^{\mathring{stop}}_{ac} \right) \tag{14}$$

where we add the losses for start and stop predictions over the context paragraph. **Proper masking must be applied** in order not to count predictions

4.9 Implementation

We have implemented our models and necessary infrastructure in `Python`, making an extensive use of `tensorflow` and `numpy`. We implement **padding** of questions and contexts shorter than the maximum sizes. We also implement **dropout** in order to provide regularization and increase generalization of the model. Our matrices are initialized using the **Xavier** initialization where applicable. To prevent gradient explosions, we **clip gradients** at a maximum global norm value and rescale them accordingly using `tensorflow` functions. However, practically, gradient clipping was not needed beyond a few initial batches. We also made extensive use of the `tensorflow.einsum` function that allowed us to write very general summations between multidimensional matrices.

5 Experiments

We experimented with several different architectures that we tried to build on our own by implementing various combinations of LSTMs and attention-like mechanisms. These attempts were relatively unsuccessful. In the end, we converged to two models:

1. **Baseline mode**
2. **Multi-perspective context matching model**

The details of their architecture are described in Section 3 and Section 4 respectively.

In our project, we exclusively used GloVe vectors of size 100 and vocabulary size 115,613. We initially trained them as well, though discontinued doing so as it proved to be very slow and not sufficiently beneficial for the loss decrease. A very easy extension of our project would be increasing the size of the word embeddings and the size of the vocabulary, reducing the number of unknown words.

5.1 Overfitting on a small dataset

For a new version of the model, we verified its functionality by overfitting on a very small dataset, running the code locally on a our laptops. This provided a guarantee that the model is actually capable of learning.

5.2 Baseline experiments

The baseline model did not perform well and the best score we achieved on the **developmental** set was F1 = 7.8 % and EM = 1.798 %. Its training was most likely hindered by a set of masking bugs in the model that we discovered. The final configuration of the baseline model was learning rate 0.02, batch size 200, dropout 0.1, maximum context length 250, maximum question length 20, hidden state size 75. The model had 995, 750 trainable parameters and its learning curve is shown in Figure 4.

Despite the number of parameters being much higher than our multi-perspective model, the performance was significantly worse. During training, the typical size of the global norm of the gradient was between 0.5 and 4.0, varying wildly. The epoch took ≈ 2000 s to run on the NVidia Tesla M60 GPU.

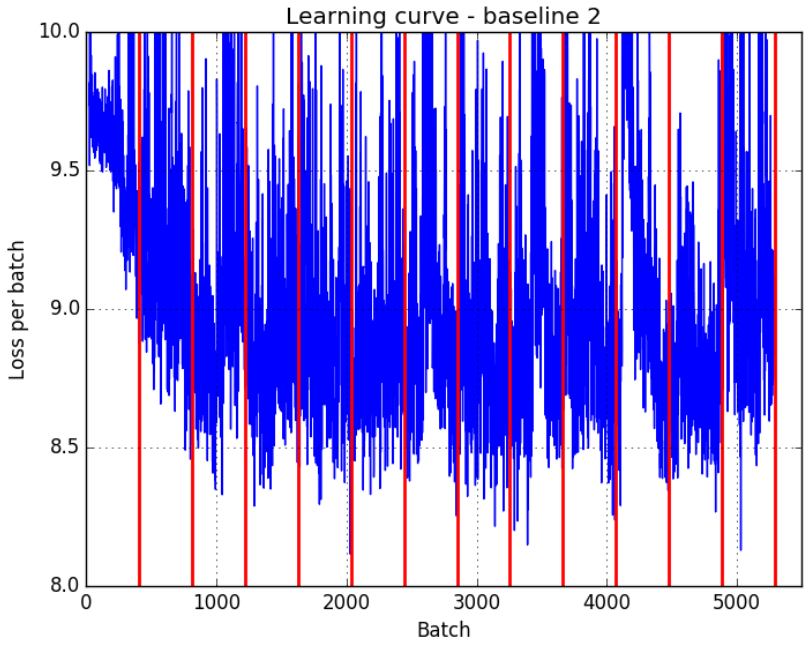

Figure 4: Baseline learning curve. The learning curve shows a very suboptimal model and learning. This plot is shown for comparison with the learning of our more advanced model. The epoch are separated by red vertical lines.

5.3 Multi-perspective matching experiments

We performed numerous experiments on the hyperparameters of our advanced model - the multi-perspective matching. We experimented with learning rates, dropouts, numbers of perspectives, and sizes of hidden layers. The best results came from the model with **only 173,882 parameters** – an order of magnitude lower than the baseline model. This was a significantly smaller model than the one implemented in the original paper [1].

The models parameters were as follows: initial LSTM hidden size 80, final LSTM hidden size 50, 13 perspectives, maximum context length 300, maximum question length 22, batch size 100, learning rate 0.01 and dropout 0.1. We extensively experimented with larger models, however, the training was very slow and the GPU was sometimes running out of memory, as the Multi-perspective matching involves large four-dimensional matrices and operations on them.

Our training is summarized in Figure 5. The plot shows a blue learning curve that was generating by training with dropout 0.1 for 7 full epochs. Each epoch took $\approx 6000\,\mathrm{s}$ on the NVidia Tesla M60 GPU. After the model started plateauing, we set the dropout to 0.0 in order to further boost the performance and continued training. The loss suddenly dropped as the model **effectively gained 10 % additional pretrained parameters**. With the zero dropout, we continued training for another 3 epochs, during which our F1 and EM on the developmental set were still increasing.

The full evolution of the F1 and EM scores on the unseen developmental set are shown in Figure 6 and Figure 7 respectively. On an unseen test dataset, we achieved **F1 = 47.516 % and EM = 34.249 %**, after 7 epoch of training with dropout 0.1 and 3 with 0 dropout, as shown in Figure 5.

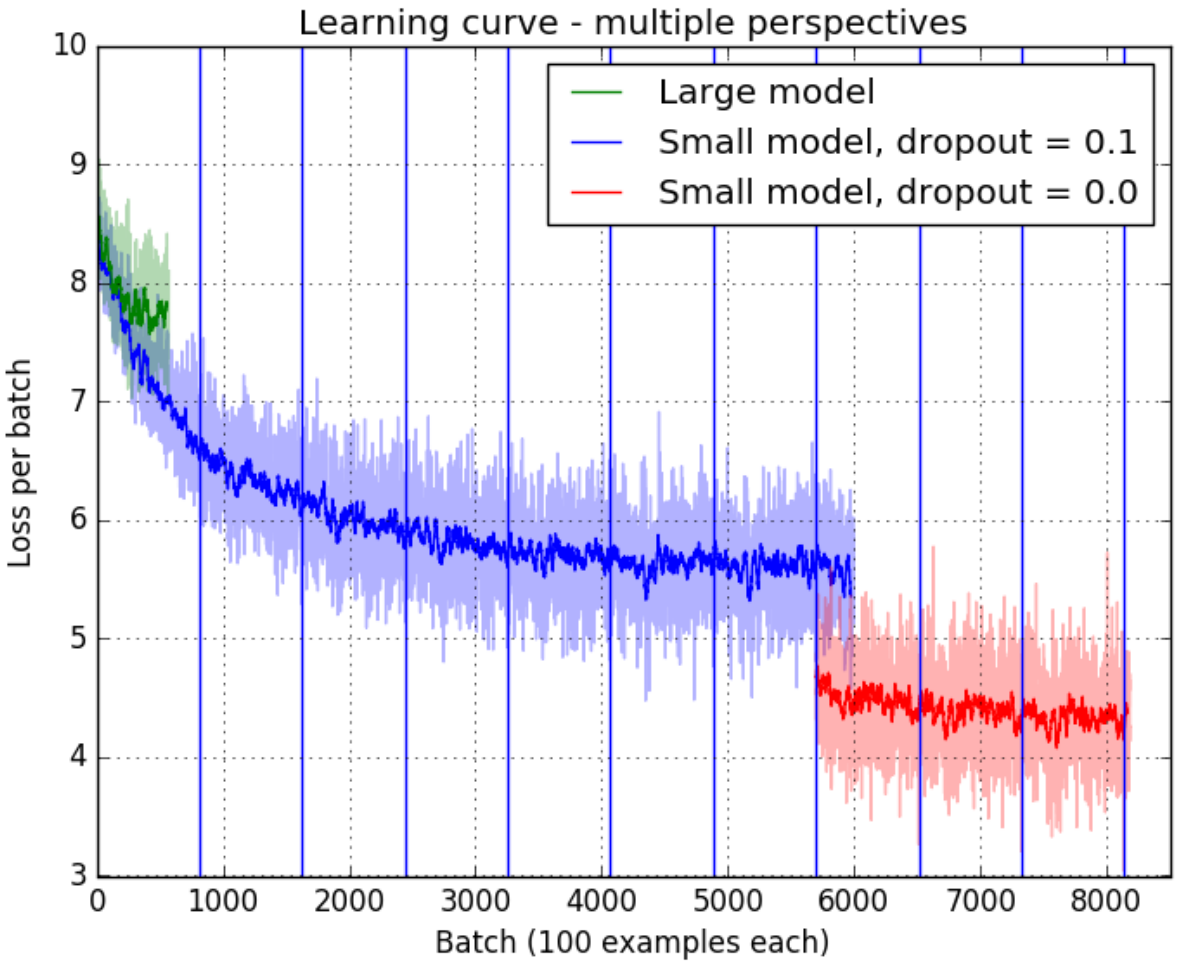

Figure 5: Learning curve for the multi-perspective matching model. Epoch are separated by the vertical lines. The blue curve is a our model with dropout 0.1 trained until epoch 7, where it starts plateauing. To boost performance, we set dropout to 0.0 and continued training, shown in red. That further boosted our performance. The green curve shows a learning curve for a larger model that started plateauing very early and was stopped.

6 Predictions

A good way to visualize the output of the model is to show the probability distribution over words of the context for the start and end of the answer. We present three examples that illustrate the behavior of our model.

6.1 Example 1 – multiple occurrences of trigger words

Question: Which hotel did the Broncos use for Super Bowl 50?
Context: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the **Santa Clara Marriott**.
Answer: Santa Clara Marriott

The probabilities for different starts and ends of the answer are shown in Figure 8. This case illustrates that the network is aware of the surroundings of words. The word *Marriott* triggers

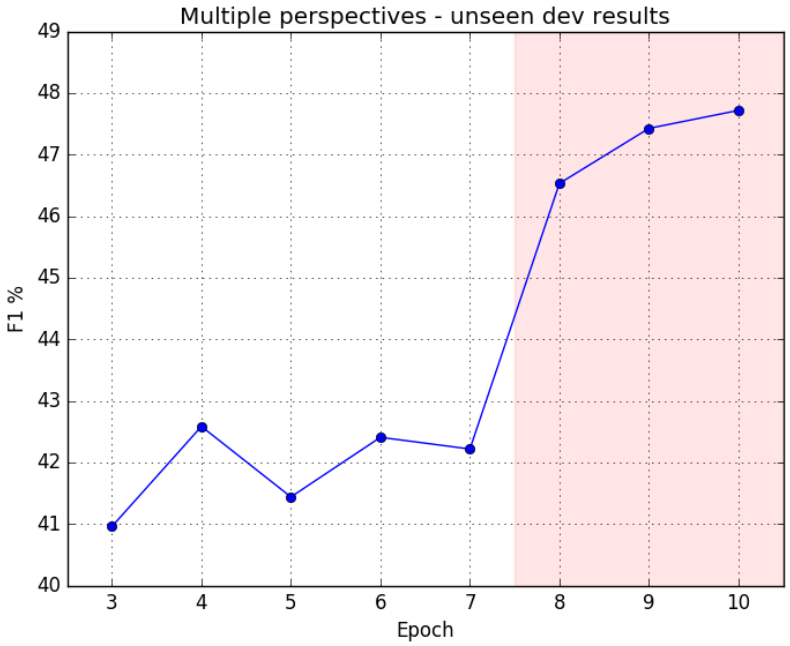

Figure 6: The evolution of the F1 score on an unseen developmental dataset for our multi-perspective model during training. The red shaded region corresponds to the epochs during which dropout was set to 0.

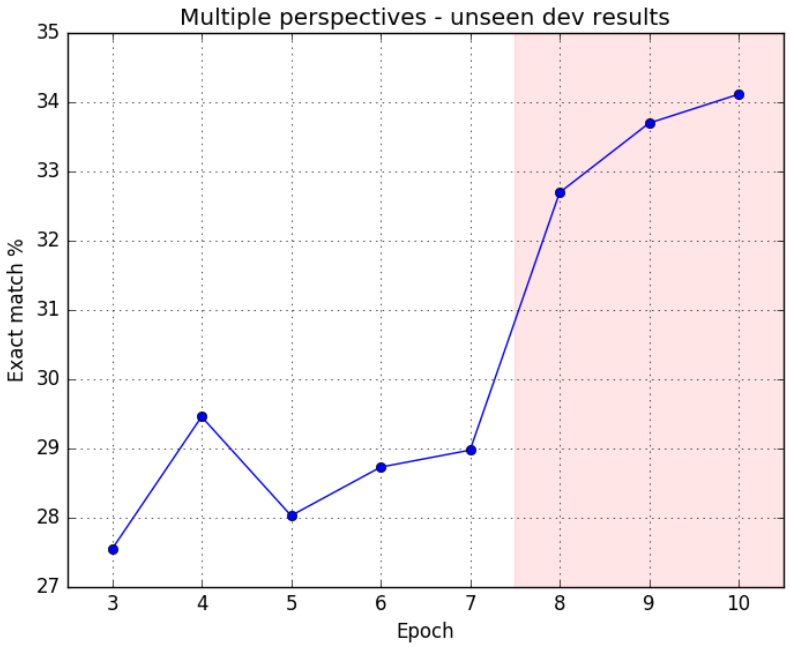

Figure 7: The evolution of the EM (exact matching) score on an unseen developmental dataset for our multi-perspective model during training. The red shaded region corresponds to the epochs during which dropout was set to 0.

the end prediction even outside of the answer, though with much lower probability. The word *San* triggers the start prediction, as it is close in the embedding space and usage to the word *Santa*, which actually is the start of the answer. It is clear that the model pays attention to

the surroundings of a word.

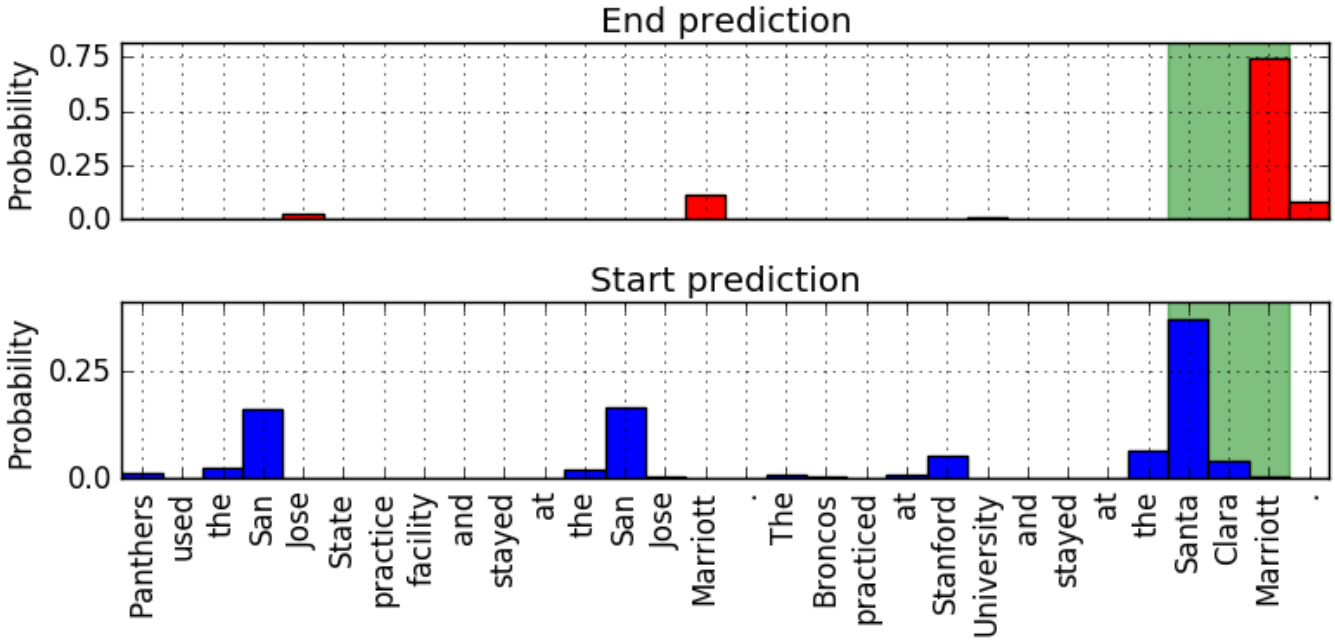

Figure 8: Probability that particular words in the context paragraph are the start and end of the answer.

6.2 Example 2 – semantic proximity

Question: How long would the astronauts be project to be able to stay on the moon for in the latter missions?
Context: The contracted batch of 15 Saturn UNK were enough for lunar landing missions through Apollo 20 . NASA publicized a preliminary list of eight more planned landing sites , with plans to increase the mass of the UNK and LM for the last five missions , along with the payload capacity of the Saturn V. These final missions would combine the I and J types in the 1967 list , allowing the CMP to operate a package of lunar orbital sensors and cameras while his companions were on the surface , and allowing them to stay on the Moon for **over three days** . These missions would also carry the Lunar Roving Vehicle (UNK) increasing the exploration area and allowing televised UNK of the LM . Also , the Block II spacesuit was revised for the extended missions to allow greater flexibility and visibility for driving the UNK .
Answer: over three days

Figure 9 shows the resulting probabilities. The answer *over three days* has the highest probability, however, the alternative *three days* is also comparable. The two answers are very close semantically and *three days* is not strictly wrong. Therefore this example shows that the resulting probabilities retain the semantic proximity of related answers. The answer itself spans the segment between the maximum probability for start and the maximum probability for end. The semantic proximity of the answers reflects in the probabilities.

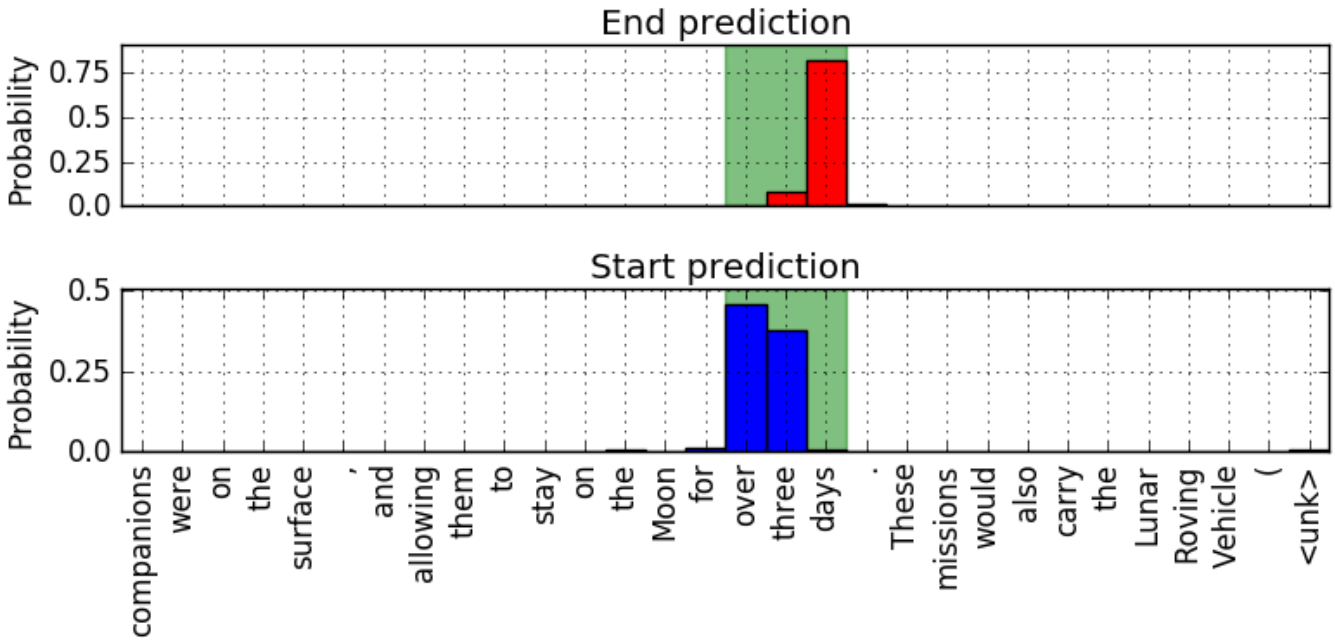

Figure 9: Probability that particular words in the context paragraph are the start and end of the answer.

6.3 Example 3 – a complicated answer

Question: What is partially responsible for weakened immune response in older individuals ?
Context: It is conjectured that **a progressive decline in hormone levels with age** is partially responsible for weakened immune responses in aging individuals . Conversely , some hormones are regulated by the immune system , notably thyroid hormone activity . The UNK decline in immune function is also related to decreasing vitamin D levels in the elderly . As people age , two things happen that negatively affect their vitamin D levels . First , they stay indoors more due to decreased activity levels . This means that they get less sun and therefore produce less UNK via UVB radiation . Second , as a person ages the skin becomes less adept at producing vitamin D .
Answer: a progressive decline in hormone levels with age

The context paragraph is quite technical and so is the question. The answer is also almost a full sentence. The probabilities shown in Figure 10 show that there is a quite a lot of potential variability in the answer. A very comparably probable answer would be *progressive decline in hormone levels with age*, with the omission of the leading particle *a*. This aligns very well with its semantic irrelevance to the meaning of the answer. Another possibility would be *a progressive decline in hormone levels* which also makes sense. This example shows that the model has an understanding of the meaning of the text, and the high probability answers are all meaningful.

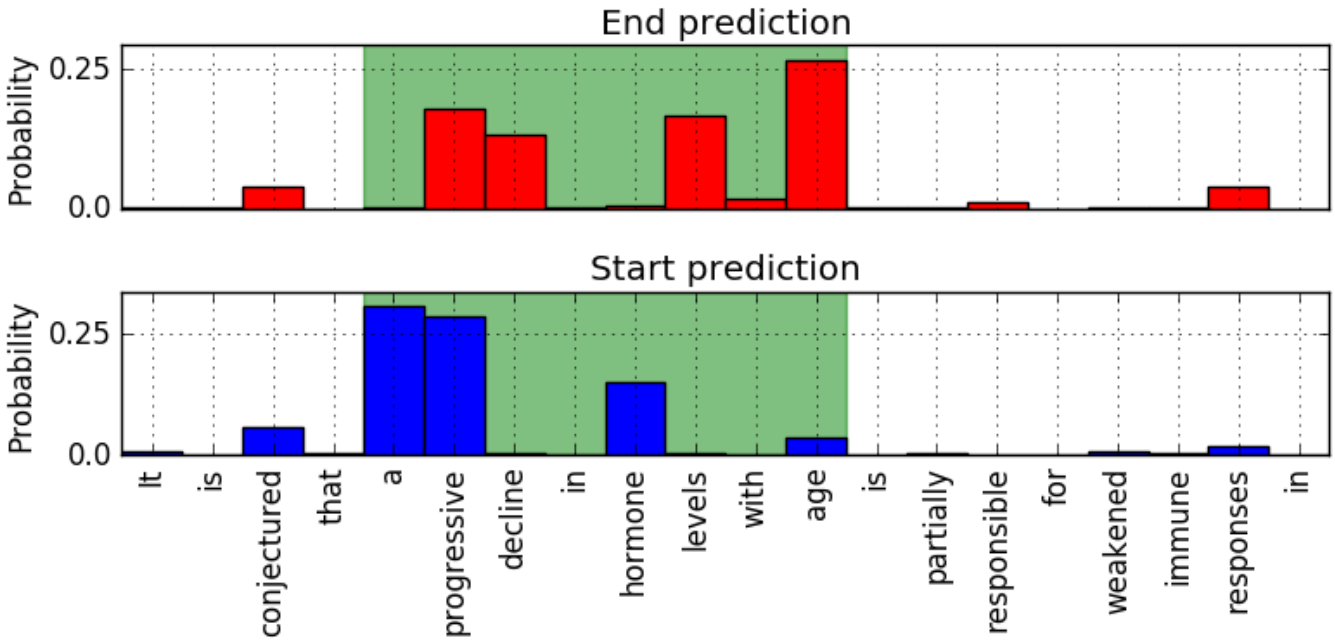

Figure 10: Probability that particular words in the context paragraph are the start and end of the answer.

7 Discussion and Results

We implemented a simple baseline model and a multi-perspective matching model for question answering on the SQuAD dataset. We performed a lot of model building and testing, exploring the hyperparameter space of our models. In the end, we were able to achieve **F1 = 47.516 %** and **EM = 34.249 %** on an unseen test set with relatively small multi-perspective matching model of only 173,882 parameters. The model significantly outperformed our baseline model, despite the baseline having an order of magnitude more trainable parameters.

We originally used the incorrect GloVe file when evaluating our answer on the development and test set, as we did not see the announcement correcting them. Using the correct word vectors increased performance and explained some of the anomalies that we saw earlier.

To further boost our performance, we trained our model with dropout 0.1 until the learning curve plateaued and then resumed training with zero dropout. This effectively increased the number of parameters in the system by 10 % and significantly lifted our F1 and EM. With more time, we would certainly explore different word embedding and dictionary sizes.

The analysis of the probabilities over context paragraphs showed that the trained model is well aligned with human intuition for what the answers should be. Looking at the next most probable candidates, one often sees answers that are very close in terms of information content and not only superficially.

8 Acknowledgements

We thank **Allen Nie**, a TA for the class, for his advice. We built our model using `tensorflow` and extensively used its documentation. We thank the CS224n team for their starter code.

References

[1] Wang Z., Mi H., Hamza W.,Florian R., *Multi-Perspective Context Matching for Machine Comprehension*, ArXiv 1612.0421