Multiple Turn Comprehension for the Bi-Directional Attention Flow Model

Thomas Liu
Stanford University
tcliu@stanford.edu
Codalab: tcliu

Abstract

Machine comprehension is a challenging and important problem in natural language processing. Attention mechanisms have recently become a popular approach for machine comprehension. In this paper, we extend an existing attentive model with multiple turn comprehension, the idea that re-reading a passage improves comprehension. Experimental results show that this extension offers promising results.

1 Introduction

An advanced question answering system is one of the ultimate goals of natural language processing and artificial intelligence research since to some extent, all NLP tasks can be rephrased as question answering problems. Semantic and syntactic understanding, reading comprehension and knowledge storage and retrieval are just a few of the language processing tasks that such a question answering system would have to handle. Application of advanced neural network techniques have led to significant progress towards that goal, aided by the availability of large and high-quality datasets.

One such dataset is the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) which frames machine comprehension as a question answering task. The dataset consists of context, question and answer triples. Context consists of paragraphs excerpted from Wikipedia articles and question and answer are generated by crowdsourced workers after reading the context paragraph. Because of the size of the dataset and the rich variety of questions and answers, the dataset has proven popular for training machine comprehension systems.

In this paper, we explore and propose extensions to the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2017), an established model that scores well on the SQuAD leaderboard. This model was chosen as a starting point for two reasons. First, it is a comparatively simpler model, in part due to the use of non-dynamic attention (i.e. the attention vector at a given time step is not a function of the previous attention vector). Second, the model's generous use of standard bidirectional Long Short-Term Memory (BiLSTM) (Hochreiter & Schmidhuber, 1997) layers makes the model very modular, allowing for easy experimentation. Different components and layer outputs can be rearranged and reused without having to worry about dimensional mismatch once the intermediate result is normalized by passing it through a BiLSTM. We begin by implementing and evaluating a baseline model that is a simplified version of the model as it is presented in the original paper. Next, we identify differences between our implementation and the original authors' implementation and quantify the impact on the model's performance. Finally, we propose improvements to the model based on architectures implemented by other competitive models.

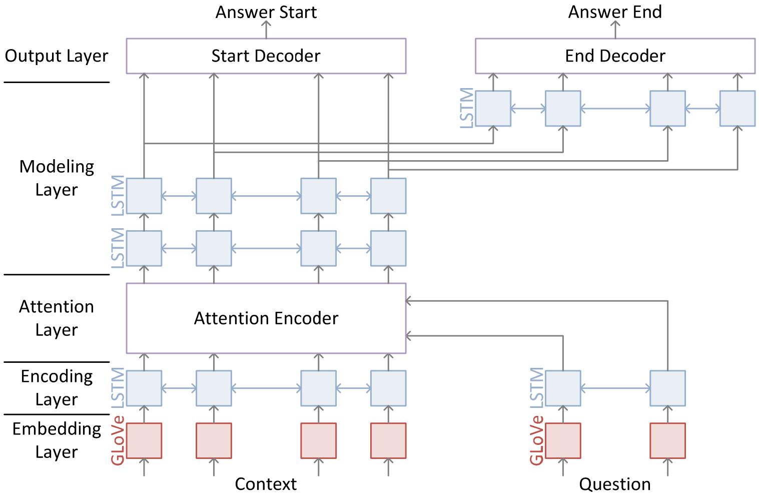

Figure 1: Bi-Directional Attention Flow Model Architecture

2 Approach

The Bi-Directional Attention Flow model implemented in this paper consists of five layers (Figure 1).

2.1 Word embedding layer

The word embedding layer converts each word in the context and question into a dense vector word representation. We use the pre-trained GloVe (Pennington et al., 2014) vectors for this layer, in particular the Common Crawl 840B tokens, 300d vectors.

The embedding layer in the original implementation also featured a CNN character embedding layer as well as a 2-layer Highway Network (Srivastava et al., 2015) that merges the two representations. In our implementation, we choose to omit the character embedding layer to reduce model complexity. Doing so is expected to result in a decrease in model performance; however, the effect is minimal based on the ablation study in the original paper. Once the character embedding layer is removed, keeping the highway layer adds no value; at best, it learns to carry the word vectors unmodified and at worst it adds noise to the word vectors.

We limit context length and question length to 300 words and 25 words respectively to reduce the size of the model. Only 1.7% of context paragraphs and 0.4% of questions in the training data set have lengths that exceed these thresholds.

2.2 Encoding layer

The outputs of the word embedding layer are fed into a BiLSTM layer which encodes discrete word vector representations into continuous context and question representations.

2.3 Attention layer

The attention layer uses the question representation to identify words in the context representation that will be useful in predicting the answer. Unlike dynamic attention implementations, the attention vector at any given time step does not depend on previous attention state. More specifically, the context and question representations combine to form a similarity matrix $\mathbf{S} \in \mathbb{R}^{C \times Q}$ where C and Q are the context and question lengths, respectively. Each element $\mathbf{S}_{i,j}$ is a function of only the ith context and jth question representation. Two sets of attention encodings are generated from the similarity matrix: context-to-question attention which selects question words important to the context and question-to-context attention which selects context words important to the question. Finally, the attention layer outputs the concatenation of the original context encoding with the two attention encodings. For more information, please refer to the original paper; the exact mathematical formulations for the attention encodings are unchanged from the ones described there.

2.4 Modeling layer

The modeling layer combines the context encoding with the attention encoding to produce output encodings that will be used to predict the answer start and end positions. The attention layer and modeling layer together can be thought of as performing a transformation similar to dynamic attention. Having separate layers allows the attention layer to focus on interactions between context and question while the modeling layer focuses on interactions between time steps.

Two layers of BiLSTMs transform the question-aware context encoding into the answer start output encoding. A third BiLSTM takes the answer start output encoding and transforms it into an answer end encoding.

2.5 Output layer

The output layer decodes the answer start and end encodings into answer start and end labels predictions. The decoder itself is a dot product between the encoding and a learned weight vector to produce logits for each context word. Taking the argmax of the logits produces the final predictions.

2.6 Loss

We define loss as the softmax cross-entropy loss between the predicted and true answer start and end labels, averaged across the batch.

2.7 Regularization

We apply a dropout rate of 0.2 to the output of every BiLSTM in the model.

2.8 Training

We use the Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.01, β_1 of 0.9 and β_2 of 0.999. Each epoch takes 1 hour to train on a single GTX 1080 GPU. The model is trained for a maximum of 10 epochs, with convergence occurring around the fifth or sixth epoch.

3 Experiments

We begin by implementing a baseline model based on our reading of the original paper with the simplifications described in the previous section. As can be expected, the baseline's performance is far short of the performance achieved in the paper. The authors have generously provided a reference implementation, and the first phase of our experiments deals with identifying key differences between the two that help close the gap in performance. The second phase involves extensions to the attention layer based on models implemented in other works, as well as changes to the handling of datasets.

3.1 Baseline improvements

From the myriad of differences between our baseline and the reference implementation, we identify three that led to a significant increase in performance.

3.1.1 Shared weights in encoding layer

Initially, we train separate weights for the question and answer encoding BiLSTMs. Sharing weights between the two encoders led to a large improvement in performance, which came as a bit of a surprise. We hypothesize that it reduces overfitting in a manner similar to dropout; with independent weights the BiLSTMs learn features specific to context and question whereas with shared weights they learn features that are common to context and question that help the attention flow layer.

3.1.2 Exponential masking

Very few context paragraphs and questions have lengths that exceed the maximum of 300 and 25 words respectively, but batch training requires that we pad these sequences to the maximum length. In general, this is problematic for training because start and end probabilities for words past the end of the context paragraph can have nonzero values, forcing the network to learn to predict zero probability for pad tokens when it should only focus on words within the context.

This is an even bigger problem in the BiDAF model because it uses softmax to generate attention encoding weights. The inputs to the attention layer are the output of BiLSTMs which already zeros outputs at nonexistent time steps. When logits for tokens at valid time steps are small numbers close to zero, logits of zero for invalid time steps are unnecessarily confusing.

We identify two different ways to mask out these logits. The first is to add a very negative number (e.g. -1e30) to invalid time steps. The second is to use a conditional operator to select between the original logits and a very negative constant. We found that both approaches perform identically. The latter has the benefit of nulling gradients when softmax is used outside of cross-entropy loss, but in this case the BiLSTM's masking also has the same effect.

3.1.3 Fixed embeddings

For the baseline implementation, we update word embeddings as part of the training process. Fixing the word embeddings resulted in a sizeable bump in performance. One possible explanation is that there are three classes of words: common words that are universally used, rare words seen infrequently in the training set (i.e. once per context paragraph) and rare words that are found only in the val, dev or test set. Assume that the rare words are generally important for predicting the answer. Training embeddings leads to moderate updates for the second class and no updates for the third class. Thus, the model performs well on the train set, but on other sets where rare word embeddings have not been trained for the task, it performs poorly.

Experiment	Dev F1	Dev EM
Baseline	15.6	10.3
+ Shared LSTM Weights	36.3	24.9
+ Logit Masking	42.3	30.2
+ Fixed Embeddings ("Gold")	**52.4**	**40.9**

Table 1: Results of Baseline Improvements

3.2 Attention layer architecture improvements

In this section, we look to other well-performing models for inspiration on how to improve the BiDAF model. Since many of the other models have similar encoder, decoder and output layers (i.e. BiLSTMs) we focus on the attention layer.

3.2.1 "Double" attention

Our first attempt at an improved attention layer takes inspiration from the Bilateral Multi-Perspective Matching model. The high-level architecture of this model is similar to that of the BiDAF model, with the main difference being that an attended question representation is also fed into the output layer. Generating an attended question representation is straightforward with the BiDAF model; simply swapping context and question inputs will generate an output that has similar dimensionality as the question. Using the attended question representation is more challenging. We could propagate it all the way to the output layer, but doing so would require a complete redesign of the output layer as the original implementation generates logits directly from the output encoding. The alternative is to encode the attended output using a BiLSTM and use it as the input to the original attention module, which we opt to do in this experiment (Figure 2a).

In practice, this approach performs poorly. We hypothesize that this arrangement causes a significant amount information about the context to be mixed into the attended question representation, reducing the effectiveness of the following attention block because it receives insufficient information about the question.

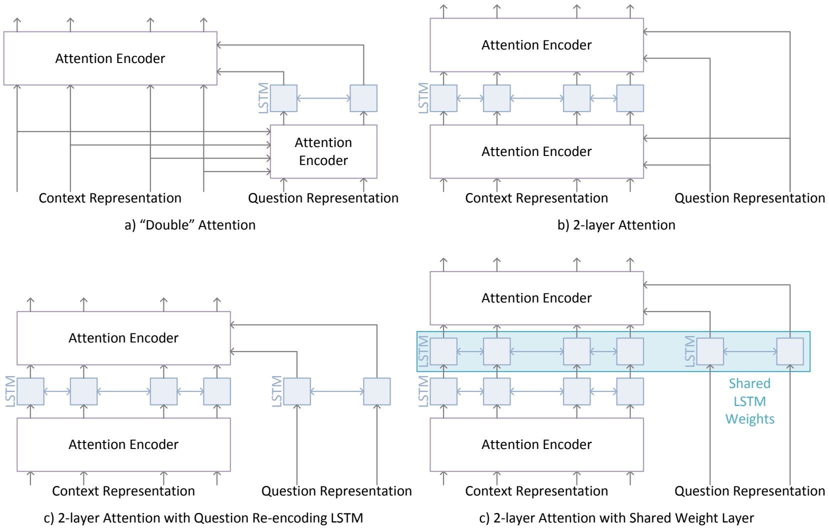

Figure 2: Proposed Modifications to Attention Layer

3.2.2 Two-layer attention

The second attempt takes inspiration from the ReasoNet model, which obtains good results by using a multiple turn approach where the context attends the question repeatedly until the level of confidence in the answer is high enough. This mimics human readers tackling reading comprehension tasks, who commonly re-read passages and/or questions several times before deciding on the answer. Our adaptation of this idea is to take the output of the original attention module, re-encode it using a BiLSTM, then use the output of the BiLSTM as the input of another copy of the attention module (Figure 2b). We use the same question encoding for both attention modules. This approach appears to have some potential, as it improves F1 and EM by several percentage points each.

3.2.3 Two-layer attention with question re-encoding BiLSTM

If the context representation has a BiLSTM between the two attention modules, should the question representation also have an additional BiLSTM as well? After all, the second attention module should be attending on different features than the first, and a re-encoding BiLSTM for the question allows the representation to be transformed into one that is more meaningful for the second attention module. This reasoning is the basis for the third modification (Figure 2c). Even though it sounds like it a solid improvement, the actual change is negligible at best.

3.2.4 Two-layer attention with shared weights BiLSTM

Looking back at the baseline experiments, we observed that sharing weights between context and question encoding BiLSTMs helped regularize the two encodings somewhat, leading to improved performance. In theory, that same trick should work here as well. For this modification, we add another context BiLSTM between the two attention modules and have it share weights with the question BiLSTM added in the previous modification (Figure 2d). Again, sharing weights has a positive benefit and both F1 and EM increase by several percentage points.

Experiment	Dev F1	Dev EM
Gold	52.4	40.9
“Double” Attention	23.1	15.9
Two-layer Attention	55.2	44.1
2-layer with question re-encoding LSTM	55.4	44.4
2-layer with shared weights LSTM (“Final”)	**59.6**	**48.6**

Table 2: Results of Attention Layer Architecture Improvements

3.3 Better handling of GloVe and SQuAD datasets

We now shift gears from experimenting with different architectures to easier changes in how the dataset is handled. When evaluating on the dev set, we notice that many words are out-of-vocabulary (OOV), i.e. not encountered in the training or validation datasets. By default, such words are mapped to a special unknown token (<unk>). However, there can be dozens of unique OOV words in a context paragraph that map to this token, which may cause the model to perform worse in this scenario. We would like our model to at a minimum be able to distinguish between different OOV words.

3.3.1 Random initialization of OOV words

We first try assigning random initializations to each OOV word encountered in the dev set. By doing so we claim that the sematic meaning of such words is not very important, but if the word appears in both context and question then the attention layer can make use of this to improve its predictions. This change leads to a small increase in overall performance.

3.2.3 GloVe initialization of OOV words

Next, we revisit the claim that semantic meaning is not important. For question answering, this is clearly not the case. Looking at the dev set, we find that many of the OOV words have corresponding word vectors in the full GloVe dataset. As before, we randomly initialize word vectors for OOV words, but if the word has a corresponding vector in the GloVe dataset, we replace the random vector with the actual one. Again, this change leads to a small increase in overall performance.

Experiment	Dev F1	Dev EM
Final	59.6	48.6
Random OOV initialization	61.6	50.5
GloVe + random OOV initialization	**62.8**	**51.9**

Table 3: Results of Dataset Processing Improvements

3.4 Final F1/EM score

After discussing with a TA at the poster session, I realized the training rate may have been too high the whole time. Lowering it to 0.001 resulted in a final dev set F1 of 71.1 and EM of 60.4. On the test set we get F1 of 71.5 and EM of 61.0.

4 Analysis and visualization

To help us understand the predictions (and more importantly, the errors) our model is making, we analyze and visualize prediction results on the dev set.

4.1 Error analysis

Of 100 randomly selected questions, 55 were answered correctly (Exact Match). Our model performs well when questions are simple and answers are unambiguous. Often in such cases, the question shares a significant amount of parallelism with the context and the answer is located close to the corresponding section of the context. Our model also performs well when the answer is numeric or is a date.

The remaining 45 incorrect answers can be categorized into six classes of errors. A third of the errors occur because the model is unable to fully comprehend either the context or the question. Another third occurs either because the model is imprecise in defining the start or end of the span, or the model incorporates far too much text into the answer span. The remaining errors are caused by the model selecting spans at random, questions requiring external knowledge for proper comprehension, or errors caused by punctuation left over from inadequate preprocessing of the SQuAD dataset. Appendix A contains examples of each class of error.

4.2 Visualization

To visualize the predictions our model is making, we graph the start and end probabilities for each word in the context. In addition, we graph probabilities in the attention layer similarity matrix **S.**

Figure 3 shows the results for a correctly answered question. The model predicts “Roger Goodell” with >90% confidence. The first similarity matrix shows strong correspondence between common phrases in both the input and question such as “early 2012” and “NFL Commissioner.” The word “Who” in the question attends on context words the most, with the answer words having the highest probabilities in that row. Curiously, the second attention layer appears to be doing something other than attention, which is something that deserves more attention.

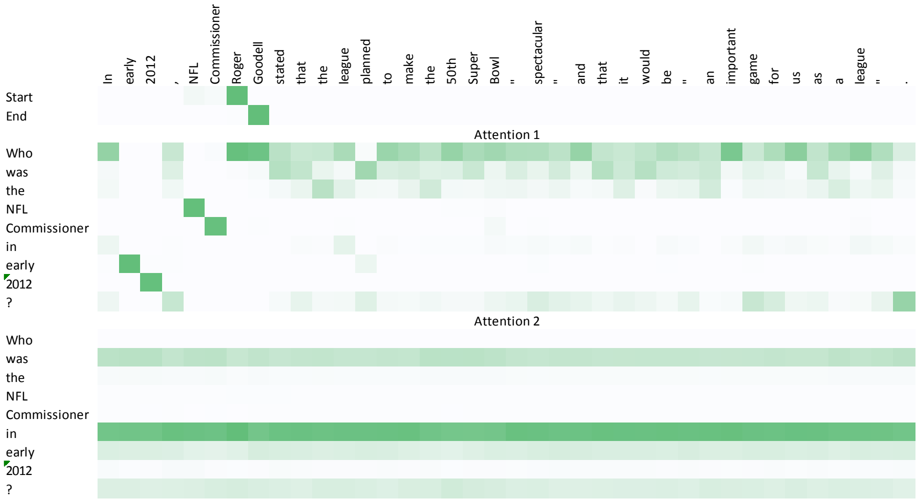

Figure 3: Probabilities for correct prediction. Softmax on attention matrices is across column

Figure 4 shows the results for an incorrectly answered question. The model’s confidence in its prediction is much lower at <40%. Again, we see strong attention between matching phrases between the context and question. More words in the question are attending on the context, but none of them exhibit the high probabilities that led the previous question to the correct answer.

5 Conclusion

In this work, we develop an attentive question answering model for the Stanford Question Answering Dataset. We create a baseline model from our reading of the Bi-Directional Attention Flow model, a model that performs well on the SQuAD task. We find that there are many small details that prevent our baseline from performing as expected and address them in our gold model. We use the gold model as a starting point for experimenting with improvements to the attention layer, ultimately deciding on an approach that uses a second copy of the attention layer and a shared weights BiLSTM to implement multiple turn comprehension. We evaluate our improvements on the SQuAD dev set, finding that multiple turn comprehension is a strong candidate for improving the BiDAF model. Our model achieves performance that exceeds the SQuAD baseline and comes close to the single model BiDAF results.

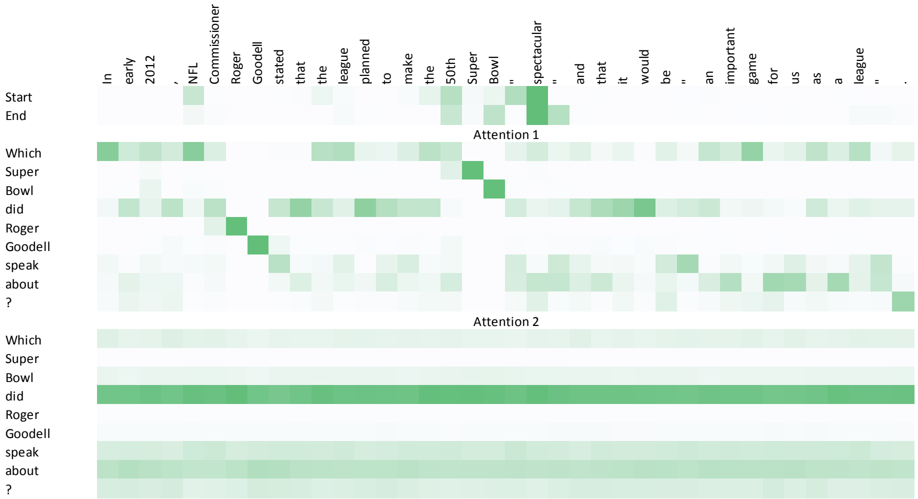

Figure 4: Probabilities for incorrect answer. Softmax on attention matrices is across column.

One topic that is severely neglected in this work is hyperparameter tuning. The BiDAF paper provided a reasonable starting value for most hyperparameters, so we deprioritized hyperparameters in favor of architectural improvements. In retrospect, there is possibly a significant amount of unrealized performance hidden in learning rate, dropout rate, state size and the like. Any future work should start with hyperparameter search.

There is still more that can be done to improve the BiDAF model. Our work only touches upon the attention layer, and future work should address other layers. Many suggestions can be readily found in the original BiDAF paper.

References

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603, 2016.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. ReasoNet: Learning to Stop Reading in Machine Comprehension. *CoRR*, abs/1609.05284, 2016.

Richard Socher Pennington, Jeffrey and Christopher D. Manning. Glove: Global vectors for word representation. *In Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 2014.

Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. *arXiv:1702.03814 [cs.AI]*, 2017.

Rupesh Kumar Srivastava, Klaus Greff, and Jurgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

Diedrik P. Kingma, and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

Appendix A: Error analysis

The following sections give details and examples of the types of incorrect predictions made by our model.

Unsuccessful comprehension: 38%

Errors in this category occur when the model is unable to fully comprehend the context and/or question. The predicted span may be a reasonable noun phrase, but is clearly wrong if both context and question are fully comprehended. The predicted span may be one of several similarly structured possible answers, but is incorrect given the context.

Context: The basic pay for a starting teacher is 27,814 p.a., rising incrementally to 53,423 for a teacher with 25 years service.

Question: What can a teacher with 25 years of experience make, in Euros?

Answer: 53,423

Prediction: 27,814

Imprecise boundary: 24%

The predicted span overlaps the true answer significantly but not fully. Often such answers are semantically correct but do not match any of the true answer spans.

Context: These advances led to the development of a layered model of the Earth, with a crust and lithosphere on top, the mantle below (separated within itself by seismic discontinuities at 410 and 660 kilometers), and the outer core and inner core below that.

Question: In the layered model of the Earth, the outermost layer is what?

Answer: crust and lithosphere

Prediction: a crust and lithosphere on top.

Span too long: 16%

The predicted span contains far too many excess words.

Context: However, as a result of the referendum in France and the referendum in the Netherlands, the 2004 Treaty establishing a Constitution for Europe never came into force.

Question: Which caused the reform to never come into force?

Answer: the referendum in France and the referendum in the Netherlands

Prediction: referendum in France and the referendum in the Netherlands, the 2004 Treaty establishing a Constitution for Europe

Random span or no span: 16%

The predicted span has no relevance to the question or is empty.

Context: Light bulbs within 100 feet of the lab glowed even when turned off.

Question: What happened to nearby light bulbs?

Answer: glowed even when turned off

Prediction:

External knowledge: 4%

A correct prediction of the answer requires knowledge not present in the context paragraph.

Context: Kenya is a presidential representative democratic republic. The President is both the head of state and head of government, and of a multi-party system. Executive power is exercised by the government. Legislative power is vested in both the government and the

National Assembly and the Senate. The Judiciary is independent of the executive and the legislature.

Question: Which branch is independant of the other branches?

Answer: Judiciary

Prediction: National Assembly and the Senate

Answering this question requires knowledge that executive, legislative and judiciary are branches of government. This is not stated in the context.

Preprocessing: 2%

The error is caused by edge cases in tokenization.

Context: The next major step occurred when James Watt developed (1763–1775) an improved version of Newcomen's engine, with a separate condenser.

Question: When did Watt finish the development of his improvements to Newcomen's engine?

Answer: 1775

Prediction: next major step occurred

1763–1775 is out of vocabulary. Removing the hyphen would lead to a correct prediction.