Machine Solver for Physics Word Problems

Megan Leszczynski
Cornell University
Ithaca, NY 14853 USA
mel255@cornell.edu

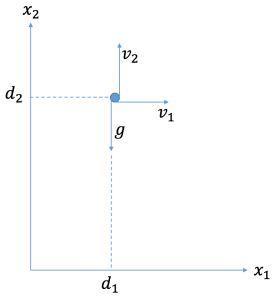
José Moreira
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598 USA
jmoreira@us.ibm.com

Abstract

We built a machine solver for physics word problems involving the free falling motion of an object under constant acceleration of gravity. Our solver consists of two long short-term memory recurrent neural networks and a numerical integrator. The *labeler* neural network labels each word of the problem, identifying the physical parameters and the question part of the problem. The *classifier* neural network identifies what is being asked in the question. Using the information extracted by both networks, a numerical integrator computes the solution. We observe that the classifier is resilient to errors made by the labeler, which does a better job of identifying the physics parameters than the question. Training, validation and test sets of problems are generated from a grammar, with validation and test problems structurally different from the training problems. The overall accuracy of the solver on the test cases is 99.8%.

1 Problem Specification

The machine solves word problems involving the physical system of Figure 1(a). Movement of the particle starts at time $t = 0$, with an initial position defined by (d_1, d_2) and initial velocity (v_1, v_2). The time behavior of the particle can be represented by the dynamical system shown in Figure 1(b). The state vector $\vec{x}(t) = [x_1(t), x_2(t), \dot{x}_1(t), \dot{x}_2(t)]^T$ consists of two positions and two velocities and its derivative depends only on itself and the acceleration of gravity. Combined with the initial condition $\vec{x}(0) = [d_1, d_2, v_1, v_2]^T$, the differential equation produces a unique solution.



(a)

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \ddot{x}_1(t) \\ \ddot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix}$$

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ \dot{x}_1(0) \\ \dot{x}_2(0) \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ v_1 \\ v_2 \end{bmatrix}$$

(b)

Figure 1: Two-dimensional space with a free falling particle (a). Displacements d_1 and d_2 define the initial position of the particle; v_1 and v_2 define its initial velocity. Gravity produces a constant downward acceleration g. Behavior of the particle is defined by the dynamical system shown in (b).

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Machine Solver

The top-level system block diagram is shown in Figure 2. The data flow through the labeler and classifier neural networks is shown in Figure 3. We used TensorFlow[TM1] to develop the neural network models for both labeler and the classifier [1].

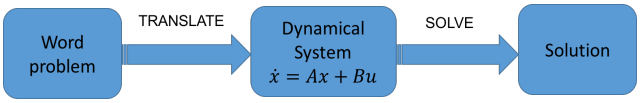


Figure 2: The first step from word problem to dynamical system is accomplished via neural networks. The second step from dynamical system to solution is achieved with a numerical integrator.

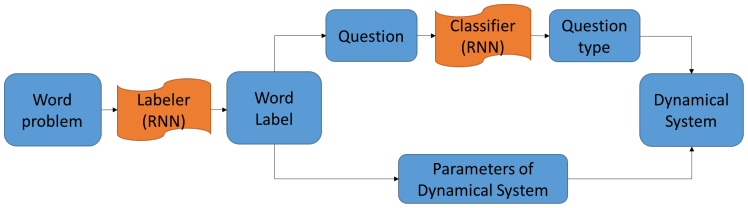


Figure 3: The word problem passes through two RNNs to produce the dynamical system form.

The labeler is an LSTM network with one hidden layer of ten units, following the model described in [2], Chapter 10, Figure 10.3. Figure 4 shows an example of the data flow through the labeler. The input to the labeler is the full problem statement and the output is a label for each word, identifying the parameters to the dynamical system and the question part of the problem. The words are input into the labeler via an embedding that is randomly initialized and trained simultaneously with the weights and biases. The weights are also randomly initialized and the biases are initialized to zero. To limit the exploration of the parameter space, we set the dimension of the embedding to equal the number of hidden units. We trained the labeler with TensorFlow's Adam Optimizer [4], an initial learning rate of 0.1, and a mini-batch size of 100 word problems. (The network architecture and hyperparameters were chosen after a limited grid search.)

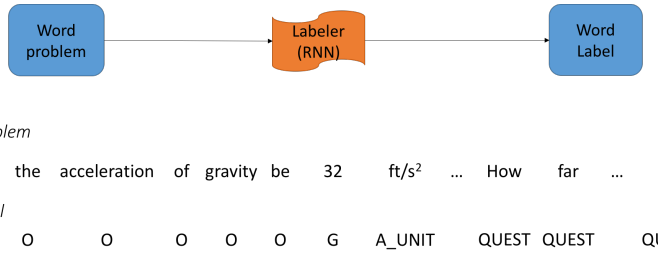


Figure 4: Example of input to labeler with expected output. A label is associated with each word, where **O** indicates other, or a word not needed for the dynamical system translation.

The classifier is an LSTM network with one hidden layer of 1,000 units. An example of the data flow through the classifier is shown in Figure 5. Only the question part of the word problem is run through the classifier. As with the labeler, we encode the words of the sequence into word embeddings, matching the dimension of the word embedding to the number of hidden units, and training them with the weights and biases. Unlike the labeler, there is only one output, the type of question, for each sequence. For more information see Chapter 10, figure 10.5 of [2] for an illustration. The classifier is trained with TensorFlow's Gradient Descent Optimizer, an initial learning rate of 0.5, and a mini-batch size of 100 questions. A grid search was used to choose these hyperparameters.

The numerical integrator computes the evolution over time of the dynamical system shown in Figure 1(b). As input it receives the initial conditions, the value of g, and the type of question

[1]TensorFlow is a trade mark of Google Inc.

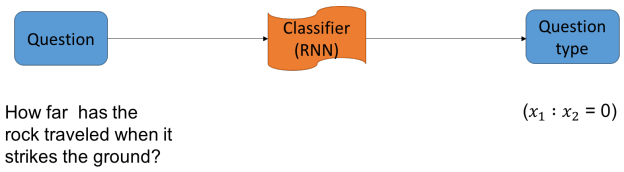


Figure 5: Example of input to classifier with expected output. Symbol x_1 refers to horizontal displacement and symbol x_2 refers to vertical displacement.

extracted from the labeler and the classifier. Using SciPy's ordinary differential equation integrator, a table of values representing the system's state to the point that the object hits the ground is iteratively constructed. The numerical solution is refined to a precision of 0.001 (one part in a thousand), based on the type of the question. For example, if the question is about the maximum height, we produce a first instance of the table, find the maximum height in that table, and then search for the maximum around that value with increased precision, repeating until we reach the desired precision. Finally, the question type is used to determine which value from the table to output from the solver.

The word problems in the training, validation, and test sets are automatically generated following a grammar. The grammar allows for mixed units (SI and US customary). The grammar also permits the initial conditions to be exposed in multiple ways. For instance, an angle value and speed will be provided in some problems, from which the solver would need to calculate the initial vertical velocity, whereas in other problems the vertical velocity can be stated explicitly. Using mixed units and various ways of providing information about each initial condition allows us to increase the complexity of the problems within the scope of the dynamical system. The grammar also ensures that the training set is disjointed from the validation and test sets, particularly in structure. This is vital in assessing the ability of the trained networks to generalize. Figure 6 shows examples of generated problems.

Assume the acceleration due to gravity is 85 ft/s^2. A ping pong ball is dropped from the top of a 8 story building, where each story is 89 m. What is the maximum speed the ping pong ball obtains?
A chair is launched at a speed of 51 mph and an angle from the horizontal of 28 degrees. Let the acceleration due to gravity on Planet Watson be 98 m/s^2. How much time has passed when it reaches its maximum height?

Figure 6: Examples of generated problems following the grammar.

3 Experimental Results

The datasets consisted of 7,000 word problems for training, 2,000 word problems for validation, and 1,000 word problems for test. The labeler and classifier reached 100% accuracy on the training set by the end of the first epoch. The epoch was broken down into fractions as the training accuracy was evaluated every seven mini-batches of 100 problems.

The accuracy on the test set after the labeler and classifier have been independently trained are shown in Table 1. The accuracy of the combined RNN system amounts to an overall accuracy of 99.8%. The labeler achieves 100% accuracy on predicting the non-question labels and incurs a small error on predicting the beginning and end of the question. As a result, the question that is extracted based on the labeler's predictions does not always match the true question. However, based on the classifier's accuracy of 99.8%, the classifier is often resilient to the errors that labeler makes in extracting the question. While the labeler incorrectly extracts ninety-one questions, the classifier only incorrectly classifies two questions from a test set of 1,000 word problems. See Figure 7 for examples.

4 Conclusions

We have developed a machine solver for word problems on the physics of a free falling object in two-dimensional space with constant acceleration of gravity. The solver has three main components.

Table 1: Accuracy on the test set of word problems. The combined RNN system accuracy is based on correctly identifying all dynamical system parameters and question type.

Labeler				
Overall	Non-question	Question	Classifier	Combined RNN System
0.909	1.000	0.909	0.998	0.998

(1) Labeler input: Let the acceleration due to gravity on Planet Watson be 65 ft/s^2. A ping pong ball is released from the top of a 3 story building, where each story is 79 m. What is the maximum speed the ping pong ball obtains?

Labeler output / classifier input: What is the maximum speed the
Classifier output: (speed : max)
Expected output: (speed : max)

(2) Labeler input: Assume the acceleration due to gravity is 49 m/s^2. A ping pong ball is launched at a speed of 35 m/s and an elevation of 88 degrees. What is the magnitude of the velocity of the ping pong ball just before it touches the ground?

Labeler output / classifier input: What is the magnitude of the velocity of the
Classifier output: (speed : max)
Expected output: (speed : x_2=0)

Figure 7: Examples of incorrectly extracted questions from the labeler and the classifier's response to them. In both cases, the question is cut short. The classifier still makes the correct the classification for the first case, but fails for the second case.

The *labeler* labels each word of the problem to identify the parameters of a canonical dynamical system that describes the time evolution of the object, and the part of the problem that corresponds to the question being asked. The *classifier* classifies the question part. Finally, an integrator is used to solve the dynamical system, producing a numerical answer to the problem.

A grammar-based generator is used to produce the training, validation and test set of problems for the neural networks. The grammar is specified so that the validation and test problems are structurally different from the training problems. We used a total of 10,000 generated problems, partitioned into 7,000 for training, 2,000 for validation and 1,000 for testing.

Both neural networks achieved almost perfect training (as measured in accuracy with the training set) by the end of the first epoch of training. When measured against the test set of 1,000 problems, the dynamical system parameters are correctly identified in all of them. The question part is precisely identified in 909 cases, but because the classifier can work with partial questions, in the end all but two questions are classified correctly. Therefore, the combined accuracy of the two neural networks, for the purpose of solving the physics problems, is 99.8%. We did have to do a systematic search over the space of neural networks to find the right architecture (number of layers and units per layer) and hyperparameters (learning rate) for the networks to behave so well.

References

[1] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015. Software available from http://tensorflow.org.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press. Book available from http://www.deeplearningbook.org, 2016.

[3] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533, 2014.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.