Multitask Learning to Evaluate Whether Personal Goals are SMART

Stanford CS224N Custom Project

Neel Rao
Department of Computer Science
Stanford University
neelvrao@stanford.edu

Abstract

The goal of this project is to use Natural Language Processing to identify whether a given written goal is Specific, Measurable, and/or Time-bound. This is motivated by findings from Education research that show that the practice of writing learning goals is beneficial for cognitive and noncognitive skill development, and that these effects are increased when the goals follow the SMART framework, meaning they are Specific, Measurable, Attainable, Relevant, and Time-bound (Lawlor, 2012; Moeller et al., 2012). Because attainability and relevancy vary based on the writer, I focus on the other three facets here. I experiment with four systems: one that uses separate vanilla RNNs with linear classification heads for each task, one that uses a single vanilla RNN base that feeds into three task-specific output heads, a system with three BERT models with linear classification heads, each fine-tuned on a separate task, and finally, a single fine-tuned BERT model base with three output heads, utilizing hard parameter sharing. I also contribute a novel dataset of 1000 written personal goals, each labeled for whether they are Specific, Measurable, and/or Time-bound. From these experiments, we conclude that while the separate fine-tuned BERT models performed marginally better (F1 scores of .93, .89, and .94 on the three tasks, vs. scores of .92, .90, and .92), the single BERT base model with three output heads may be advantageous because of its parameter cost savings and reduced training time. This finding hopefully contributes to the development of AI goal-setting coaching.

1 Key Information to include

- Mentor: Anna Goldie
- External Collaborators: None
- Sharing project: This project may become a component of my Symbolic Systems Master's Thesis

2 Introduction

The goal of this project is to use NLP techniques to build a multi-task classifier that can distinguish whether students' written learning goals are Specific, Measurable, and Time-bound. This project is motivated by two findings. The first is that goal-setting as a practice in K-12 education has been shown to be instrumental in increasing student achievement, as well as developing a variety of social and emotional constructs like self-efficacy, agency, and sense of purpose (Moeller et al.,2012). The second finding is that while goal-setting in general is useful, the quality of the goals also matters, and makes the practice even more effective (Lawlor, 2012). I hope to utilize the SMART goal framework as one measure of quality. SMART is an acronym meaning Specific, Measurable, Attainable, Relevant, and Time-bound. While attainable and measurable are dependent on the author of the goal, the other three

criteria are author-agnostic, and therefore implementable given my dataset.
The ultimate aim for this project is to build a classifier that could be used in classrooms to evaluate students' learning goals. While goal-setting is becoming increasingly prioritized in education, teachers often lack the time necessary to evaluate students' goals, on top of the academic feedback they already give. This system could hopefully be used to automate that process.
To the best of my knowledge, this is a previously unattempted problem. In addition, no public dataset of personal goals exists, much less, one that is labeled for any of the characteristics of SMART goals. Because of that, a large part of this project was collecting, cleaning, and labeling a dataset of 1000 goals, with 3 labels each, for whether they are Specific, Measurable, and/or Time-bound. While only having 1 rater is suboptimal, this is not considered to be fatal, because interrater reliability among SMART goal raters is generally quite high (Lawlor, 2012).
I experimented with four systems to solve this problem, which varied across two factors. The first factor was model architecture, which was either a vanilla RNN, or a BERT Transformer. The second factor was whether the models utilized multitask learning or had separate models for each task. The multitask learning was done with hard parameter sharing, where the three tasks share the exact same base model weights, whose output is fed into three separate output layers. I experimented with the architecture because, while Transformers are generally considered the best available model, I wanted to compare them to the baseline of vanilla RNNs, to get a better understanding of their performance in context. The motivation behind experimenting with multitask learning is three-fold, and includes performance and deployment considerations. For one, it is likely that goals have some latent representations that are shared across classifications. For example, the language used in making a goal time-bound and the language used for measurability can both involve units of time or other quantitative metrics. Secondly, given the relatively small dataset, we may run the risk of overfitting to the training set. Multitask learning is able to prevent this because the combination of losses acts as a regularizer across tasks. As a final reason, given the desire to apply this in a K-12 or other relatively low-tech environment, being able to achieve the same performance with less parameters, and therefore less device memory used, may be helpful.
After running these experiments, I find that, as expected, the Transformer models perform significantly better than the vanilla RNNs, across the multitask and single-task versions. The Transformers produce F1 scores between .89 and .94, while the RNNs produce F1s between .54 and .82 (with most around .6). More interestingly, however, I find that the multitask Transformer system performs as well as the three independent single-task models. This is impactful because it offers a path to significant savings in the number of parameters and time spent training the model, while keeping performance pretty high.

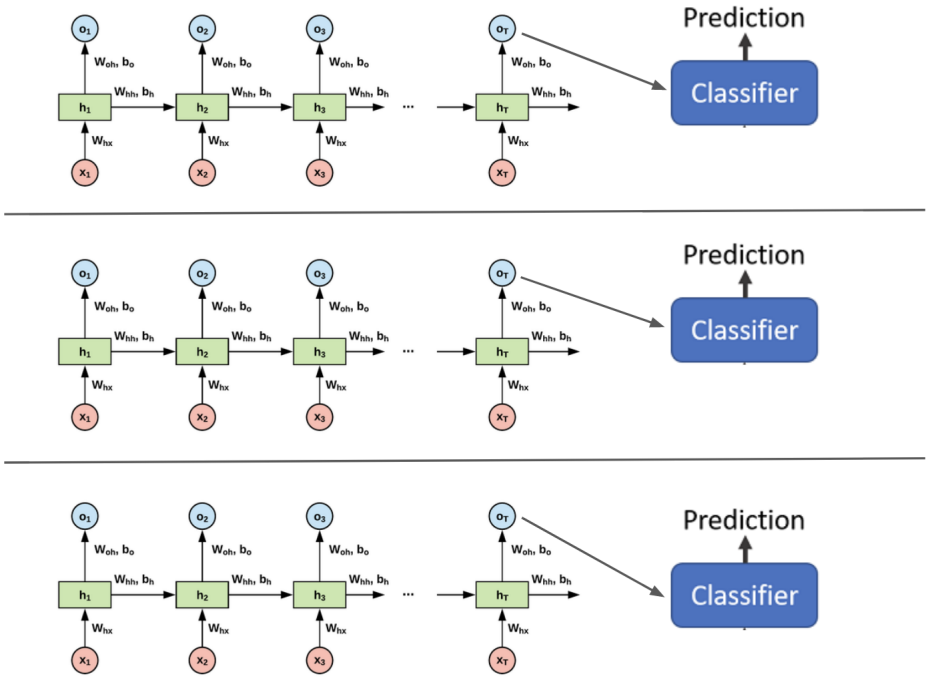
3 Related Work

There is little related work on this task itself, using natural language modeling to evaluate SMART goals. However, there is a good deal of work on the methodologies employed. Fine-tuning a BERT base model that takes in a sequence of text and then feeds it into a final linear output layer for binary classification is a common method, and many of my initial hyperparameters come from other papers (Devlin et al., 2019). In addition, multitask learning is also a common area of exploration in the field. It usually comes in two forms–either hard parameter sharing, where the tasks share a common base model, and only vary their output layers, and soft parameter sharing, where they each have completely different architectures, but their parameters are kept close to each other through regularization. The benefits of each are numerous, including forming better, more robust representations, and reducing the likelihood of overfitting (Ruder 2017). For this task, I use hard parameter sharing, to reduce the total number of parameters in the system.

4 Approach

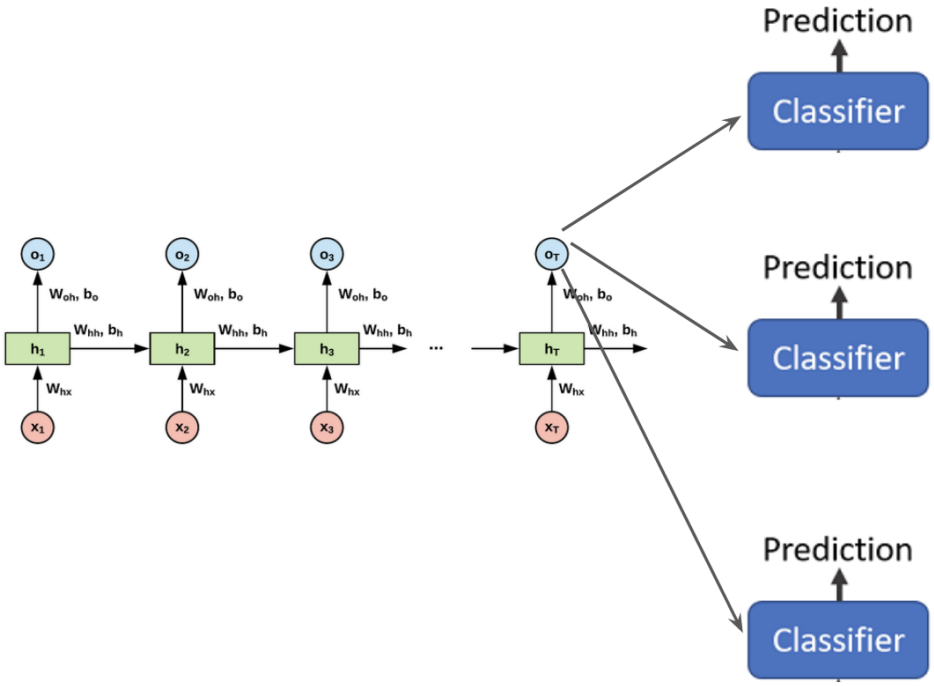
As mentioned above, I ran four experiments, with four different systems:

System 1



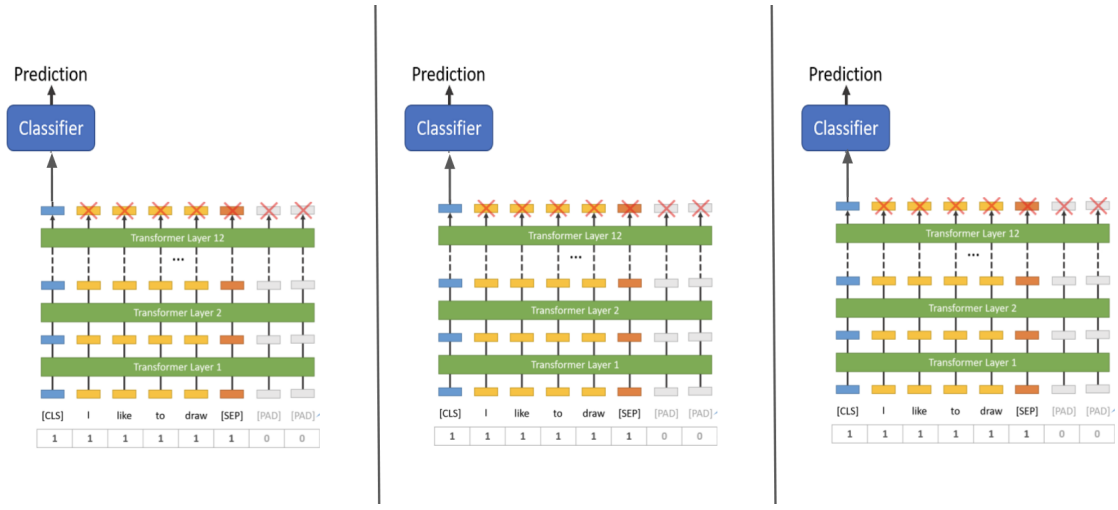
The first system contains three separate models, each a simple vanilla one-layer RNN, with embedding size 50, and hidden layer dimensionality of 256. Each performs binary classification on one of the three labels of interest. They are each trained independently, and do not share weights. However, they each utilize the same pre-trained word embeddings, from the GloVe dataset. This decision was made to ensure the baseline system was as rudimentary as possible, and because the limited size of the dataset will most likely make it challenging to get significantly better results from fine-tuning these embeddings (although this is yet to be seen). The implementation of this lookup was borrowed from CS 124's Assignment 5. Lastly, each model culminates in a single sigmoid output layer, which is rounded to either 1 or 0 to make a prediction. The architecture is fairly standard, but implementation was assisted by Jake Tae's (2020) python tutorial. The results reported below are from models trained for 25 epochs, but the results did not vary greatly across different numbers of epochs. Lastly, the models utilized AdamW optimization, with a learning rate of .001. Because of the small scope of the dataset, training time was quite low, generally less than 10 minutes. The data was preprocessed and lemmatized, in part with code taken from Sheridan Stewart's Sociology 128D notebook, with modifications made for the specific content.

System 2



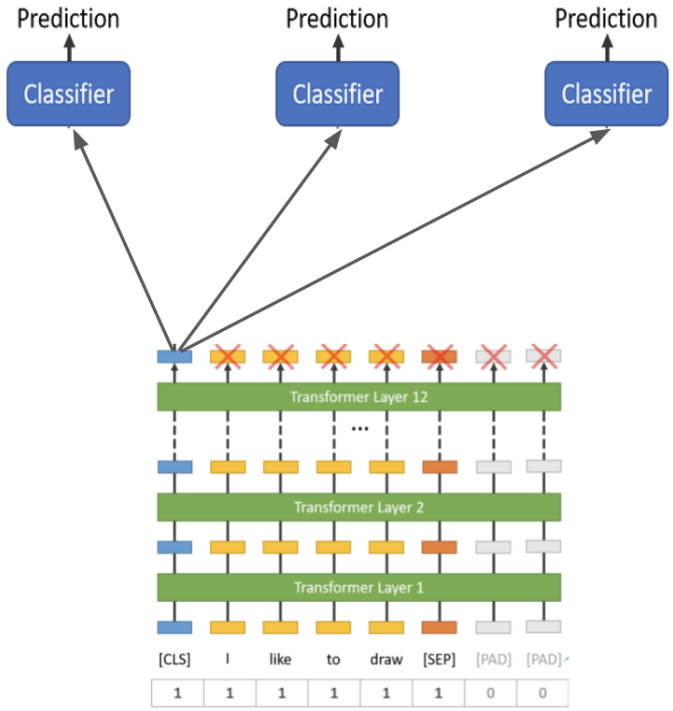
The second system also uses the vanilla RNN structure, but consists of just a single base model. The final hidden state from this base model is then inputted into three different heads, each a fully connected linear layer, with a sigmoid output, each for one of the three classification tasks. The losses on each task are used to train each final layer. The base layer is trained with the sum of the three losses. Because all three tasks are binary classification tasks, and all are of equal importance in deployment, the losses did not need to be weighted differently. The RNN structure is the same as in the first system, with embedding size of 50 and hidden size of 256, and uses the GloVe vectors, a learning rate of .001, and AdamW optimization. It also uses the same pre-processing techniques as the first system. Once again, the number of epochs did not affect performance very much, but the results shown below are from 50 epochs.

System 3



The third system utilizes three separate BERT models, each connected to a single linear output classification layer. Each model is fully fine-tuned on one of the three classification tasks. The BERT models were the BertForSequenceClassification class from the HuggingFace database, for uncased inputs. A maximum input length of 300 tokens was chosen after inspecting the distribution of token lengths, and choosing a length that would balance the need for fewer parameters with the desire to minimize the amount of input sequences with missing information. The tokenizer was a pretrained BERT uncased tokenizer. The output size of the BERT model was 768. This output was then fed into a dropout layer with probability .3 (to reduce overfitting), and then into a linear output layer and a final sigmoid function, to make the binary classification. Once again, we used an AdamW optimizer. As suggested by Devlin et al., we used a learning rate of 2e-5 and trained for 3 epochs. This training took about 2 hours for each model, training on Google Colab, for 6 hours total. Some code, especially regarding creating Dataloaders for this system and the fourth system, was taken from Venelin Valkov's (2020) online HuggingFace tutorial, and modified for this situation.

System 4



The final model is a single BERT base, whose final hidden layer is then inputted into a dropout layer with probability .3, and then into three separate final output layers, each corresponding to a single classification task. The BERT model is pretrained, but the final output layers are not. The model utilizes hard parameter sharing. All aspects of the BERT model are the same as above, and all aspects of the final linear layers are the same as the linear layers above. We used the same number of epochs and learning rate, and trained on Google Colab. Training also took 2 hours for this model, meaning it took a third of the time that the third system did.

5 Experiments

5.1 Data

The dataset was hand collected, cleaned, labeled and preprocessed. It was collected by searching the internet for goal examples, mostly hand collected, but in certain cases, scraped with some string manipulation. These were then supplemented with learning goals crowdsourced from various Stanford Computer Science students. These examples were then further cleaned in a jupyter notebook, and labeled as specific, measurable, and/or time- bound by hand by a single rater. While a lack of multiple raters is suboptimal, SMART goal evaluation has been found to have high interrater reliability, so it is believed to be sufficient (Lawlor, 2012). The dataset consists of 1000 goals, each mapped to a triple. Of these goals, 694 are labeled as specific, 473 are measurable, and 357 are time-bound. They were then split into a training set with 700 examples, and a test set with 300 examples.
In addition, care was taken to make sure that the correlations between each label was not too high. Goals that exhibited multiple SMART characteristics were sometimes chopped up such that they only exhibited one of the characteristics. For example, the goal ("I want to run 20 miles by the end of this month", which is both measurable and time-bound, might be manually changed to two goals, "I want to get better at running by the end of this month" (time-bound but not measurable) and "I want to run 20 miles" (measurable but not time-bound). While these factors do most likely correlate in the wild (e.g., a person who is trained in SMART goal-setting is likely to write a goal that fulfills all three), it is important that the model for time-boundedness is not just picking up on measurability. This could happen if the correlation between the two is too high.
For further specification of the task, it is as follows: given a written learning goal, evaluate whether it is specific, relevant, and/or time-bound. For example, given the input "Upon completion of the difficult airway workshop, participants should be able to formulate an accurate algorithm for the management of an obese adult patient with inadequate face mask ventilation", the model should output that the goal is specific (because it focuses on one specific algorithm), measurable (because we could evaluate whether the participant can formulate the algorithm or not), and time-bound, because it specifies that we want to accomplish this by the end of the workshop) (Chatterjee and Corral, 2017).

5.2 Evaluation method

The evaluation methods used are F1 score and accuracy. F1 score is preferable because of the class imbalances, especially in specificity and time-boundedness. F1 can be calculated by taking the harmonic mean of precision and recall.

5.3 Experimental details

Experimental details, on model configurations, learning rate, training time, and other factors, can be found in the Approach section.

5.4 Results

The primary metric used for analysis is F1 score. The following table describes the F1 scores for each of the 4 systems, on each of the 3 tasks. Each column represents the performance on the given task, as measured by F1 score. The highest score across systems on each task is bolded.

F1 Scores

System	Specificity Task	Measurability Task	Time-boundedness Task
System 1 (Three RNNs)	.827	.606	.538
System 2 (Single Multitask RNN)	.77	.636	.61
System 3 (Three BERT Transformers)	**.933**	.890	**.938**
System 4 (Single Multitask BERT Transformer)	.923	**.905**	.915

The BERT Transformer models significantly outperform the RNN models, as expected. They are pretrained on quite a bit of data, have many more parameters and many more layers, are much more expressive, and represent the state-of-the-art.

Within the RNN systems (Systems 1 and 2), the multitask version appears to perform a little bit better, which is encouraging for multitask learning as a whole. Its performance is especially higher on the Time-boundedness task. In addition, the RNN systems have the same rank order of performance on the three tasks, performing best on the Specificity task, followed by the Measurability task, then Time-boundedness. These results are a little surprising, because the Specificity task was actually the most difficult for the human rater, requiring the closest reading and the most cognition. Part of the increase in Specificity performance may be due to the large number of examples labeled specific, which raises the number of true positives, and can raise F1 accordingly.
Within the BERT systems, we get our most surprising result. The two systems perform almost identically, and both achieve fairly promising results, with F1 scores around .9. With a test set size of 300, the differences in performance are reflective of only a small difference in number of examples classified correctly. The similarity between the two is encouraging because it allows for a decrease in the amount of parameters necessary to get these results. This suggests that multitask learning is suited to these three tasks. This does make sense post facto, as the inputs (sequences of text) and the outputs (binary classification) are the same across all the tasks, which suggests an ability for the base BERT model to form representations that are useful across all three classifications.

The secondary metric used is accuracy, reported below. As in the above table, the highest score across systems on each task is bolded.

Accuracies

System	Specificity Task	Measurability Task	Time-boundedness Task
System 1 (Three RNNs)	70%	62%	72%
System 2 (Single Multitask RNN)	66%	60%	69%
System 3 (Three BERT Transformers)	**90**%	89%	**95**%
System 4 (Single Multitask BERT Transformer)	89%	**90**%	93%

The accuracies further support the conclusions about the Transformer systems drawn from the F1 scores. The rank orders of the BERT systems across the three tasks do not change at all. The rank orders of the RNN-based systems do change for two tasks, Measurability and Time-boundedness, however, as the multitask RNN reports a higher F1 score on these tasks but a lower accuracy. The discrepancy between F1 score and accuracy may suggest that the multitask system is more robust to class imbalance. This makes sense because of the regularization effect of its shared representations (the output of the base RNN). This is another sign that multitask learning may be promising in general for this set of tasks.

6 Analysis

An error analysis of the classification abilities of the single-task BERT models and the multitask BERT model reveal similar mistakes. In fact, because their results are so similar, it is almost impossible to distinguish between their errors. Because of this fact, in this section, I will focus on the types of mistakes the two BERT systems made on each of the three tasks. The RNNs make similar mistakes, but also make significantly more mistakes, many of which don't have a pattern decipherable to a human. The models may not be refined enough for in-depth error analysis, so I will stick to just the BERT models here.
With regards to the specifity task, the models have a difficult time with sentences where some aspect of the sentence is specific, but the action itself (and therefore the goal) is not. For example, the model returns a false positive for the goal "I will develop my relationships with David, Sarah, and Mom". While the the goal describes specific people, the verb phrase 'develop my relationships' is not specific enough. This is a particularly difficult case, and I could imagine a human who is reading quickly being easily fooled by this as well.
Regarding the measurability task, the models struggle with actions that are measurable by virtue of being binary, either accomplished or not accomplished, with no metric attached. For example, one false negatvie was the goal "get into college". It is measurable, but to know that, you'd have to understand the nature of the task. The models have to understand that the task has finite states. Usually goals that are just a few words long, with one verb, are not measurable.

The models performed best on the time-bounded task, which makes sense given that it was also the easiest for the human rater (i.e., me) to label, requiring the least reading comprehension, and mostly looking for specific phrases. The main issue that the models had was returning false negatives when the units of time were unfamiliar jargon, like in the goal "create a detailed advancement matrix in Q1". This could easily be solved by training the models on more language that included the phrases "Q1", "Q2", "Q3", and "Q4". If this were deployed in a corporate environment, the models would quickly understand the context-specific jargon.

7 Conclusion

I believe this project offers two contributions. First, it creates a dataset of 1000 written personal goals, labeled for whether they are Specific, Measurable, and/or Time-bound. Hopefully this dataset can be used for other projects similar to this one, to develop AI coaching. Secondly, this project offers an NLP system to evaluate written personal goals for specificity, measurability, and time-boundedness. This model consists of a single fine-tuned BERT base, that then branches into three task-specific linear output layers. The model achieves good (though not spectacular) results, with F1 scores and accuracies in the low .9 range across the 3 tasks. Its performance is comparable to the baseline of 3 separate Transformers, and significantly better than the performance of a multitask RNN or individual single-task RNNs. This model serves as a proof of concept for Transformer architectures, and also for multitask learning. The potential suitability of multitask learning to this problem is important because it allows for significant savings in storage size of the model, which could be useful in deployment. The results are not yet high enough for deployment, but this project points in the right direction, and provides a path forward to further explore Transformer-based multitask learning for these tasks. It shows that this set of tasks is practical, and I am hopeful that with more data, and more refinement, it can be a component of an AI coach in the future.

References

Bowman, J., Mogensen, L., Marsland, E., Lannin, N. (2015). The development, content validity and inter-rater reliability of the SMART-Goal Evaluation Method: A standardised method for evaluating clinical goals. Australian occupational therapy journal, 62(6), 420-427.

Lawlor, K. B. (2012). Smart goals: How the application of smart goals can contribute to achievement of student learning outcomes. In Developments in business simulation and experiential learning: Proceedings of the annual ABSEL conference (Vol. 39).

McCormick, Chris. "Bert Fine-Tuning Tutorial with Pytorch." BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick, 22 July 2019, https://mccormickml.com/2019/07/22/BERT-fine-tuning/.

Moeller, A. J., Theiler, J. M., Wu, C. (2012). Goal setting and student achievement: A longitudinal study. The Modern Language Journal, 96(2), 153-169.

Olah, Christopher. "Understanding LSTM Networks." Understanding LSTM Networks, Colah's Blog, 27 Aug. 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Ruder, Sebastian. "An Overview of Multi-Task Learning for Deep Learning." Sebastian Ruder, Sebastian Ruder, 24 Oct. 2018, https://ruder.io/multi-task/index.htmlfn10.

Tae, Jake. "Pytorch RNN from Scratch." Jake Tae. (2020). https://jaketae.github.io/study/pytorch-rnn/simple-rnn.

Valkov, Venelin. "Sentiment Analysis with Bert and Transformers by Hugging Face Using Pytorch and Python." Curiously, Venelin Valkov, 20 Apr. 2020, https://curiously.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/.