CALYPSO: A Neural Network Model for Natural Language Inference

Colin Man colinman@stanford.edu
Kenny Xu kenxu95@stanford.edu
Kat Gregory katg@stanford.edu

Abstract

The ability to infer meaning from text has long been regarded as one of the "benchmarks" of the quest to artificially approximate human intelligence. The field of Natural Language Inference explores this task by explicitly modeling inference relationships in natural language. In this work, we present the CALYPSO model, which builds upon Chen et al. '16's EBIM model by enhancing the matching layer with modifications to Chen's soft attention as well as three matching algorithms inspired by Wang et al. '17. Although CALYPSO's 82% accuracy is 3.2% lower than that of our EBIM implementation, our ablation study and comparison of training loss over time suggest that every modification has value and that hyperparameter tuning as well as revisions to the merging framework promise better results.

1 Introduction

Natural Language Inference (NLI) attempts solve problems with two main inputs: a premise and a hypothesis. Given a *premise*, which is known to be true, and a *hypothesis* whose veracity is unknown, an NLI algorithm classifies the relationship between the two sentences as one of three classes, determining whether the premise *entails*, *contradicts*, or is *neutral* to the hypothesis. For example, the premise "some wolves eat deer" implies the hypothesis that "some animals eat deer" since "wolves" is a hyponym of "animals" and is thus an *entailment*. However, the same premise would not imply the hypothesis "some birds eat deer" and is a *contradiction*.

The creation of the SNLI dataset in 2015 (described below) spurred a wave of rapid innovation in the NLI field. As of March 2017, 26 different papers have been published reporting accuracy on this SNLI dataset. The current state of the art models are Chen et al. '16, which achieved 88.6% accuracy, and Wang et al. '17, which attained 88.8% accuracy.

In this paper, we present CALYPSO, a Neural Network Model for NLI that builds on Chen's EBIM model by exploring and including other approaches to attention and matching inspired by Wang's Bilateral Multi-Perspective Matching model.

2 Background

2.1 Data

We work with the Stanford Natural Language Inference (SNLI) Corpus, which contains 570K English sentence pairs written and labeled by humans. Each premise/hypothesis pair is tagged as either entailment, contradiction, or neutral. The dataset has been verified through Mechanical Turk and ambiguous pairs have been trimmed out.

2.2 Related Work

The earliest work on SNLI dataset was done in 2015 by the dataset's creators, Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. Using a lexicalized classifier, they achieved 78.2% test accuracy. Soon after, sentence-based encoding models (ex. LSTM's) improved the known performance to 83-85$\%$ with only a couple million parameters. However, most recent work has been done with more complex models that extend different neural network frameworks on top of sentence encoders.

Currently, the two papers that perform the best are Chen et al. '16 and Wang et al. '17. Both utilize multi-layered frameworks in which both the premise and hypothesis are run through an encoder (generally BiLSTM) before a matching method is applied between the two statements and a similarity between matched words is inferred. Chen matches with soft-attention and infers differences between the premise and hypothesis with concatenation, subtraction, and element-wise multiplication. On the other hand, Wang adds multi-perspective methods and infers differences with cosine similarity. Both Chen and Wang used many more parameters (6-8 million) than in previous papers.

3 Approach

We first implement a baseline Bag-of-Words Model as described Bowman et al. 2015. Next, we implement the Enhanced BiLSTM Inference Model (EBIM) as described by Chen et al.'16, which introduces soft-attention. Afterwards, we supplement the EBIM model by incorporating three elements of Wang '17's Bilateral Multi-Perspective Matching: hard-attention, full-matching and maxpool-matching. We then perform an ablation study to determine the importance of the four components.

All models begin by translating the premise and hypothesis from words into GloVe word embeddings, and all end with a three-way softmax classifier.

3.1 Bag-of-Words Baseline (BOW)

Our baseline Bag-of-Words Model is inspired by Bowman's 100d Sum-of-Words model. We represent the premise and hypothesis as the mean of the GloVe 100d word embeddings of their component words. These two vectors are concatenated and fed into a three-layer, 200d feed forward neural network with tanh non-linearity and a softmax classifier.

3.2 Enhanced BiLSTM Inference Model (EBIM)

The Enhanced BiLSTM Inference Model implements the model described by Chen. For each statement s (premise or hypothesis), let the corresponding statement in the premise/hypothesis pair be $g(s)$. EBIM begins with the words of both statements in the pair and for each statement uses a BiLSTM to encode the word embeddings, $\mathbf{w}_s$, into a vector $\mathbf{p}_s$ in preparation for matching. $\mathbf{p}_s$ is the result of concatenation of the forward and backward hidden states from the BiLSTM at each index. Let $\mathbf{p}_{s,a}$ be the encoded vector for word a of statement s.

The matching layer is based on soft-attention. An attention matrix like the one visualized in *Figure 1* is computed between the premise and hypothesis statements by taking the exponentiation of the dot-product similarity between each word in the encoded vectors and every word in the corresponding statement:

$$\mathbf{q}_s = \mathbf{p}_s^T \mathbf{p}_{g(s)}$$
$$\mathbf{e}_s = e^{\mathbf{q}}$$

where $\mathbf{e}_{s,a,b}$ is a scalar value representing the dot product of $\mathbf{p}_{s,a}$ and $\mathbf{p}_{g(s),b}$

EBIM then takes

$$\mathbf{c}_{s,a} = \sum \mathbf{e}_{s,a,b} \mathbf{p}_{g(s),b}$$

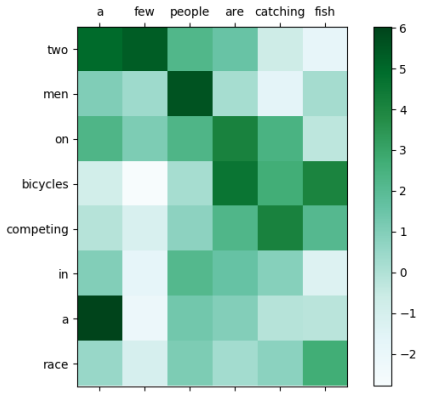

Figure 1: *Soft-Attention Matrix Example of Contradiction. Hypothesis is along x-axis and premise along y-axis.*

for each word $a \in s$ where $\mathbf{e}$ is normalized along the axis of summation, as the context vector $\mathbf{c}_s$ of the statement.

This vector is then composed through an inference step, giving the final matching output:

$$\mathbf{m}_s = [\mathbf{c}_s, \mathbf{p}_s, \mathbf{c}_s - \mathbf{p}_s, \mathbf{c}_s \circ \mathbf{p}_s]$$

where ',' denotes concatenation.

After matching, $\mathbf{m}_s$ contains alignment context information and is passed to the composition layer, which consists of another BiLSTM. This layer models the interaction between the subcomponents of each statement.

After composition, the BiLSTM outputs are passed to a pool merge layer, in which the statements are average and max pooled. The premise and hypothesis pooled outputs are then concatenated into a single, fixed-length vector. This vector is fed into a two layer feed forward network and, finally, a softmax classifier.

3.3 CALYPSO

CALYPSO uses the same model architecture as EBIM, visualized in *Figure 2*. However, during the matching phase, CALYPSO uses four different matching methods instead of Chen's single attention method to calculate $\mathbf{m}_s$ (*Figure 3*):

1. **Weighted Soft-Attention**: CALPYSO modifies the soft attention used in the matching layer of the EBIM model by adding a weight $\mathbf{W}$ to the calculation of $\mathbf{e}$:

   $$\mathbf{e} = \mathbf{p}_s^T \mathbf{W} \mathbf{p}_{g(s)}$$

   CALYPSO also removes the exponentiation step used in Chen '16 due to the introduction of hard-attention and instead uses L2 normalization.

2. **Hard-Attention**: CALYPSO uses an approach similar to soft-attention for hard-attention, but adds a step in between that "hardens" the same attention matrix by taking the encoded vector that corresponds to the max attention for each word:

   $$w = \arg_b \max \mathbf{e}_{s,a,b}$$
   $$\mathbf{c}_{s,a} = \mathbf{p}_{g(s),w}$$

   for each word $a \in s$.

3. **Full Multi-Perspective Matching**: CALYPSO uses an approach from Wang '17 to extract multi-perspective similarities between words in two statements as follows:

   $$\mathbf{c}_{s,a} = \mathbf{F}(\mathbf{W}\mathbf{p}_{s,a}, \mathbf{W}\mathbf{p}_{g(s),b})$$

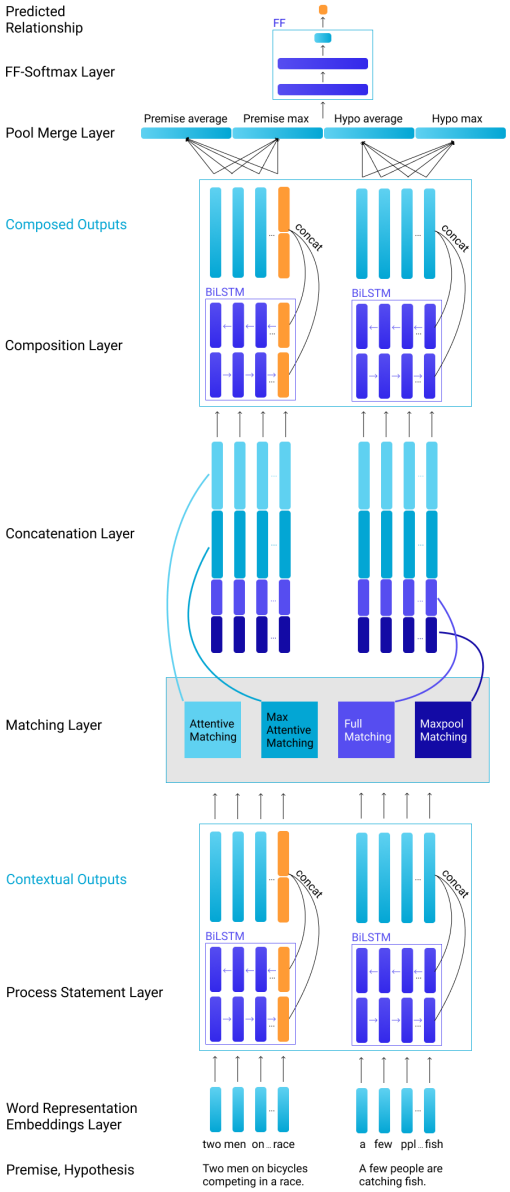

Figure 2: *CALYPSO passes contextually encoded words through the four matching methods shown in Figure 3, sends the outputs through a composition layer (BiLSTM), Pool Merge Layer (Avg/Max of Premise and Hypothesis), and 2-layer Feed-Forward Network (tanh), and finally classifies with softmax.*

where **W** denotes the weight matrix used to extract perspective dimensions, **F** is element-wise cosine similarity, and b is the last word in $g(s)$. Thus, similarities are calculated between the last state of the premise and all **p** in the hypothesis and vice versa.

4. **Maxpool Multi-Perspective Matching**: CALYPSO uses the same approach as Full Matching, but instead of using only the last state of $g(s)$ to calculate $\mathbf{c}_{s,a}$, uses the element-wise maximum of calculated similarity vectors between a and every other word in $g(s)$.

For the two Multi-Perspective Matching methods, we were forced to reduce the dimensions of **p** from 300D to 100D so we would not run out of RAM during runtime. We achieved this with a basic matrix transformation of **p** before it is fed into the Multi-Perspective Matching methods.

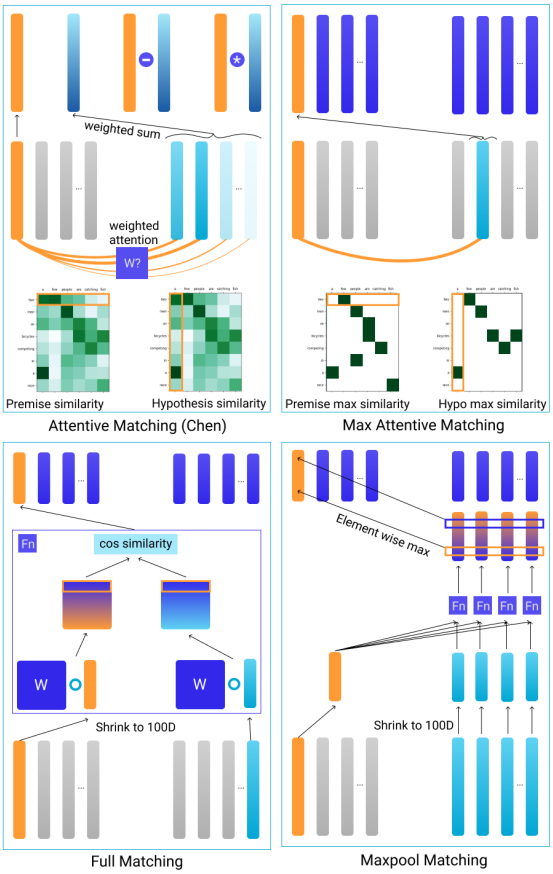

Figure 3: *Matching Methods. Attentive soft and hard matching are shown on the top. Multi-perspective "full" and "max" matching are shown at the bottom.*

When combining the various matching methods, CALYPSO concatenates the outputs $\mathbf{c}_s$ from the methods and passes them through a composition LSTM and Feed-Forward Softmax classifier with pool-merge as in the original Chen '16 model.

3.4 Other Approaches

In addition to the model used in CALYPSO, we experimented with a number of additional features that were excluded from the final model because they failed to improve performance. We tried including the original embedding vectors $\mathbf{w}_s$ in the input to the composition step. We also built a model that used multi-layered BiLSTM for processing and composition. Further, we varied our hidden layers between 100, 200, and 300 dimensions. We tested models with more feed-forward layers as well.

4 Experiments

4.1 Settings

For our baseline, we use an optimal dropout rate of 10%, a learning rate of 0.001 (Adam), and regularization lambda of .0001, determined through random grid search hyperparameter validation. For EBIM and CALYPSO, we use Chen's optimal parameters with a dropout rate of 50% and a learning rate of 0.0004 (Adam). Regularization was explored as an option but removed early on.

We train the models using cross entropy loss.

We use GloVe word vectors for our distributed word representations. BOW uses 100d vectors, while EBIM and CALYPSO use 300d vectors. We train these embeddings as we train our model. The hidden layer dimensions of the BiLSTMs are 300d; those of the final feed forward network are 100d.

We train in batches of 32 statement pairs. Each batch of statements must be padded such that all statement lengths are the same. We select batches using shuffled bucketing to reduce excess computation over padded indexes. To do so, we sort sentence pairs by length while keeping the order random between sentences of the same length and then partition this sequence into batches. As a result, batches contain similar length sentences. We shuffle the order of batches to avoid training on all shorter sentences before all longer ones.

4.2 Evaluation Metrics

We trained our models for at least ten epochs and continued until the training loss stopped improving by more than 0.03 over two consecutive epochs. We evaluated on the dev set every few epochs and tested on the model epoch that yielded the best dev accuracy.

4.3 Hyperparameter Selection

Because of time and computing limitations, we were only able to perform hyperparameter validation on our BOW baseline. We computed our optimal BOW parameters using random grid hyperparameter search over $50,000$ training examples. This initial validation also helped us verify the efficacy of our hyperparameters and caused us to remove the regularization constant when it was shown to be relatively ineffective.

We also implemented random hyperparameter search with cubic surface interpolation to better visualize the hyperparameter search space of our models. Although we used only Chen's parameters in our final experiment, *Figure 4* showcases the effectiveness of this approach for future work.

4.4 Code

To help duplicate our results, we publish our code at

`https://github.com/katgregory/nli-calypso`

5 Results and Discussion

Our implementation of EBIM based on Chen '16 produces the best results out of the three models. CALYPSO performs 3.2 percentage points worse in terms of accuracy. However, in our ablation study, we find that removing any matching method from CALYPSO further reduces our performance. This indicates that each matching method contributes to the accuracy. For example, the 9.3% discrepancy in the training and testing accuracy in the CALYPSO w/o maxpool matching ablation experiment suggests that maxpool matching may play an important role in preventing overfitting in the full CALYPSO model. This may be because the element-wise max in the maxpool matching algorithm notices large weights more than the other matching algorithms.

In addition to verifying the contribution of each method via the ablation study, we also track the gradients of parameters in all four matching methods and verify that they are updating as predicted. Our results in *Figure 5* point to two most likely conclusions: revisions to merging framework and further hyperparameter tuning.

CALYPSO merges the four different types of matching by concatenating the output and passing it to the composition BiLSTM layer. We believe that there is potential for improvement in this area.

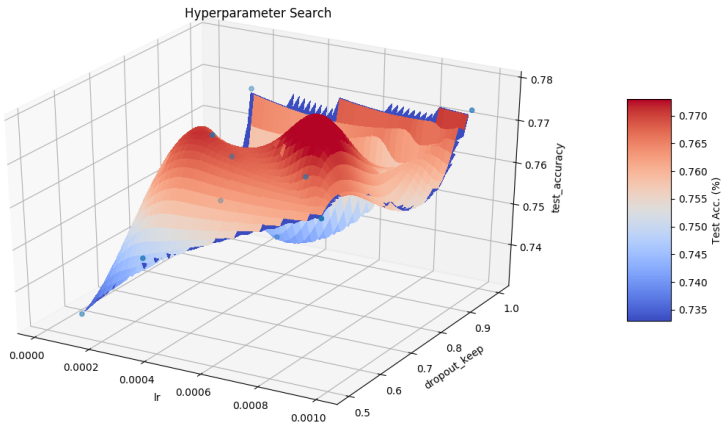

Figure 4: *Random hyperparameter search interpolation example over Chen's EBIM model + weighted attention*

Model	Train	Dev	Test
CALYPSO	86.6	82.0	**82.0**
w/o wtd. atn.	88.5	**82.2**	81.6
w/o full m.	89.5	81.9	80.7
w/o maxpool m.	**90.2**	81.9	80.9
w/o max atn. m	88.7	81.7	81.0
EBIM + wtd. atn.	93.7	85.5	84.4
EBIM (based on Chen)	**93.8**	**86.1**	**85.2**
BOW (based on Bowman)	78	-	75.6

Figure 5: *CALYPSO performance (% accuracy). Weighted (wtd.) attention uses a bilinear product of contextual outputs as weights for matching as opposed to a dot product.*

Refer to "Future Work" for more discussion.

Testing performance on CALYPSO used the same hyperparameters as Chen '16, which may not be as suitable for our model due to the number of parameters added from the additional matching algorithms. In particular, the learning rate of CALYPSO, as seen in *Figure 6*, may be too low. It is very possible that a significant improvement can be achieved if we perform validation on CALYPSO as we did with the BOW model.

6 Conclusions and Future Work

CALYPSO is unable to improve upon EBIM using the matching methods described in Wang '17. However, each matching method has value, indicating that revision to the composition layer or hyperparameters should improve accuracy.

Looking forward, we identify three areas ripe for further exploration. First, we wish to find more effective ways of combining the matching methods. Summation, subtraction, and element-wise multiplication are interesting starting points. Second, we would like to perform validation to find the optimal learning rate, dropout rate, and batch size for the CALYPSO model. We can also tune the hidden dimension sizes and number of layers in FF to an ideal level. Our own experiments use

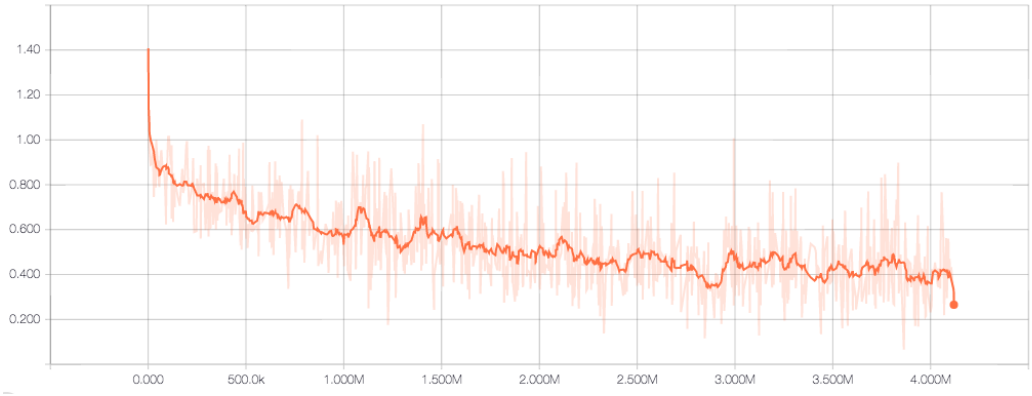

Figure 6: *Training Loss of CALYPSO over 570K training pairs and 7 epochs*

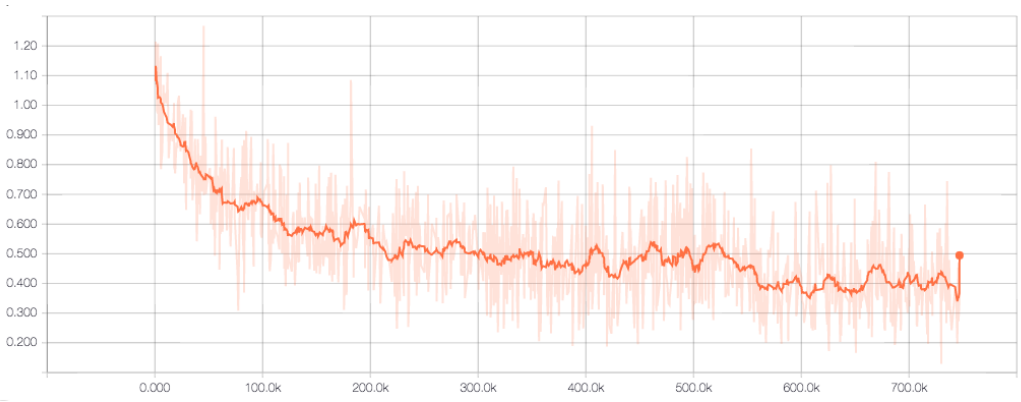

Figure 7: *Training Loss of EBIM over 570K training pairs and 1.5 epochs. EBIM reaches the same loss levels as CALYPSO much faster, indicating that there is room for CALYPSO hyperparameters to be adjusted.*

Chen's optimal parameters, which are likely suboptimal for CALYPSO given the sizable additional parameters. Third, we would explore using distinct encoding LSTMs for soft/hard attention and full/maxpool matching in order to allow each LSTM to specialize.

Acknowledgments

We would like to thank Ignacio Cases for his excellent mentorship throughout the evaluation of our project, as well as Chris Manning and Richard Socher for their instruction and inspiration.

References

[Bowman et al., 2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326, 2015.

[Chen et al., 2016] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequen- tial and tree lstm for natural language inference. arXiv preprint arXiv:1609.06038, 2016.

[Parikh et al., 2016] Ankur P Parikh, Oscar Tackstro m, Di- panjan Das, and Jakob Uszkoreit. A decomposable atten- tion model for natural language inference. arXiv preprint arXiv:1606.01933, 2016.

[Pennington et al., 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In EMNLP, 2014.

Appendix

We thought you might enjoy this snippet of our commit log.

```
| | |\
| | * | 29e0cda Beware the Jubjub bird, and shun  Ubuntu, 5 days ago
| * | | 17444e9 Beware the Jubjub bird, and shun  Ubuntu, 5 days ago
* | | | aaaa316   The jaws that bite, the claws that catch!  Kenneth Xu, 5 days ago
| |_|/
|/| |
* | |   3df0e4d Merge branch 'master' of https://github.com/katgregory/cs224n  Ubuntu, 5 days ago
|\ \ \
| |/ /
| * |   a2ed2c7 Merge branch 'master' of https://github.com/katgregory/cs224n  Ubuntu, 5 days ago
| |\ \
| | |/
| | * 9bac491 Long time the manxome foe he sought.  Kenneth Xu, 5 days ago
| | * ec956ca So rested he by the tum-tum tree  Kenneth Xu, 5 days ago
| | * 9058553 and stood a while in thought.  Kenneth Xu, 5 days ago
| | * 30514d7 And as in uffish thought he stood,  Kenneth Xu, 5 days ago
| * | 70539a8 He took his vorpal sword in hand;  Ubuntu, 5 days ago
| |/
* | d209670 the frumious bandersnatch!  Ubuntu, 5 days ago
|/
* 8fe34ff the Jabberwock, with eyes of flame,  Kenneth Xu, 5 days ago
* dad2bba Came whiffling through the tulgey wood,  Kenneth Xu, 5 days ago
* 7f456ec And burbled as it came!  Kenneth Xu, 5 days ago
* 163ffb1 One, two!  One, two! And through and through  Kenneth Xu, 5 days ago
* 6ca62a0 the vorpal blade went snicker-snack.  Kenneth Xu, 6 days ago
* 87780bd He left it - dead - and with its head,  Kenneth Xu, 6 days ago
* 18a5ca1 he came galumphing back.  Kenneth Xu, 6 days ago
* f6106c1 And hast thou slain the Jabberwock?  Kenneth Xu, 6 days ago
* fb02240 Come to my arms, my frabjous boy.  Kenneth Xu, 6 days ago
* 6dfcc05 Come to my arms, my beamish boy  Kenneth Xu, 6 days ago
* 32c194d 'Oh frabjous day- Calooh, Calay'  Kenneth Xu, 6 days ago
* 1df595a He chortled in his joy.  Kenneth Xu, 6 days ago
* e36d7c9 'Twas brillig and the slythy toves  Kenneth Xu, 6 days ago
* 5c9916e did gyre and gimble in the wabe;  Kenneth Xu, 6 days ago
* 6a90d0b All mimsy were the borogroves,  Kenneth Xu, 6 days ago
* 26766d3 and the mome raths outgrabe.  Kenneth Xu, 6 days ago
* c0b7109 and the borogroves outgrabe  Kenneth Xu, 6 days ago
* 974cdc2 raabbit hole  Kenneth Xu, 6 days ago
* 3aa6095 lskjfds  Kenneth Xu, 6 days ago
*   9c91fa1 Merge branch 'master' of https://github.com/katgregory/cs224n  Kenneth Xu, 6 days ago
|\
```