



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Deep Neural Networks

Team 2: Literary Paradox Miners

Kaggle Competition: ***Contradictory, My Dear Watson***

Jeff Simon, Yaron Walfish, Sina Azartash, Linshen Wu

Competition Description

- Can machines determine the relationships between sentences, or is that still left to humans?
- Classify pairs of sentences (a premise and a hypothesis) into three categories - entailment, contradiction, or neutral using GPU.
- Entailment (logically requires)[1]:
 - A entails B if the truth of A requires that B is true.
 - A: I will turn 37 this year, B: I am currently living, A entails B.
- Profound implications for examples such as fact-checking, identifying fake news, and analyzing text

Competition URL: <https://www.kaggle.com/c/contradictory-my-dear-watson>

Our Team

Member Duties:

Yaron Walfish: improved the model, focused on model structure

Jeff Simon: improved the model, focused on model structure

Linshen Wu: presentation editor, improved the model, focused on overfitting

Sina Azartash: presentation and research

Our Strategy:

- 1) Convert sequences of words into numbers
- 2) Create 2 separate models: one for english and one for all languages
- 3) Find the best pretrained model for our dataset
- 4) Use Transfer Learning
- 5) Used several strategies to improve our Transformer Model
- 6) Fine-tune Our Model

Final Project URLs

- English model and EDA notebook:

https://colab.research.google.com/drive/1u9a5kD0NtsGR3mItaFIT_aTXybVRe5ax

- Multi-language notebook:

https://colab.research.google.com/drive/1qiR2ImnIZ3buQXE6dnCPKpM9-RxCcRF_#scrollTo=9o8D4eUlgQs5

- Submission notebook:

https://colab.research.google.com/drive/1zTK2tK817x8ZII9HumvmN_ZoeXhkmbBP#scrollTo=9o8D4eUlgQs5

- Final presentation on Youtube:

https://www.youtube.com/watch?v=stZ_85s3Gss

Acronyms



NLI

Natural Language Inferencing



NLP

Natural Language Processing



EDA

Exploratory Data Analysis



TN

True Negative



TP

True Positive



FN

False Negative



FP

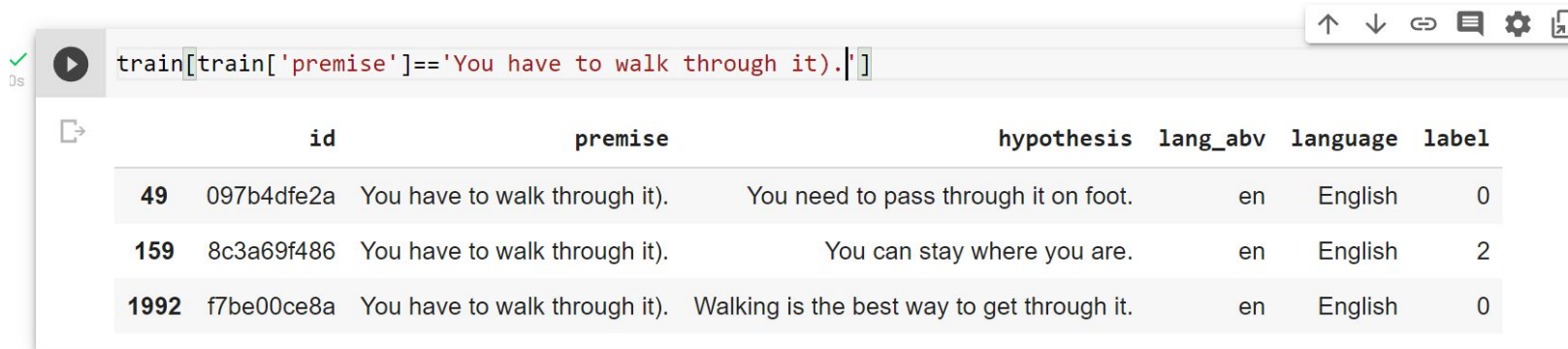
False Positive

Key Features

Input Features:

1. **Premise:** sentence 1 with 8209 unique values
2. **Hypothesis:** sentence 2 with 12119 unique values
3. **Language:** 15 different languages

Note: the same **premise** could associate with different **hypothesis** and **label** as:



```
train[train['premise']=='You have to walk through it).']
```

	id	premise	hypothesis	lang_abv	language	label
49	097b4dfe2a	You have to walk through it).	You need to pass through it on foot.	en	English	0
159	8c3a69f486	You have to walk through it).	You can stay where you are.	en	English	2
1992	f7be00ce8a	You have to walk through it).	Walking is the best way to get through it.	en	English	0

EDA



Observation:

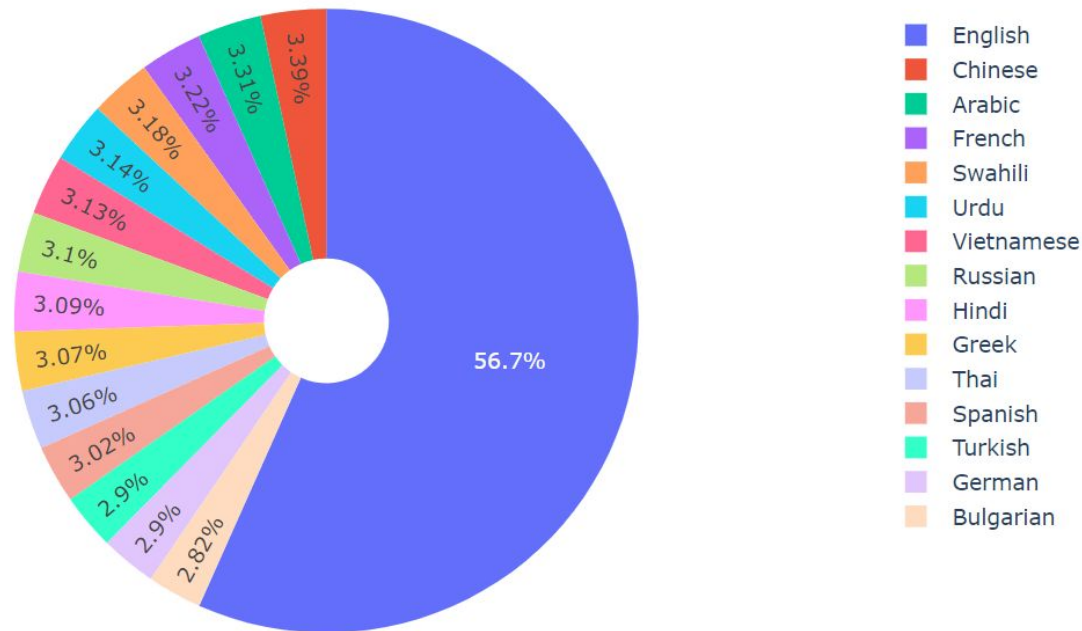
The training set contains **three** different target labels with **12120** total records as:

- **4176 Entailment**
- **4064 Contradiction**
- **3880 Neutral**

Observation on training data:

- There are total of 15 different languages
- Major language is English: 56.7%
- Rest of languages range: 2.82% ~ 3.39%

Percentage distribution of different Languages



Preprocessing

- Tokenizer: sequence of words → numerical representations
 - We are using an NLI version of Roberta. Roberta is a more specific version of BERT
 - ROBERTA uses three kinds of input data: input word IDs, input masks, and input type IDs.
 - These inputs help the tokenizer to ignore padding and distinguish the premise and hypothesis
 - For proper tokenization, pad all of the inputs to be the same size.
 - The [CLS] token denotes beginning of the input, [SEP] token denotes separation between the premise and the hypothesis.

```
1 def encode_sentence(s):  
2     tokens = list(tokenizer.tokenize(s))  
3     tokens.append('[SEP]')  
4     return tokenizer.convert_tokens_to_ids(tokens)
```

```
1 encode_sentence("I love machine learning")
```

```
[146, 16138, 21432, 26901, 102]
```

The Pretrained Model

- Two pretrained models are used for preprocessing: **roberta-large-mnli** & **xlm-roberta-large-xnli** [2]
 - Both are the same model, but use different pretrained weights
 - **Roberta-large-mnli** - english pretrained weights
 - **xlm-roberta-large-xnli** - multilanguage pretrained weights
 - Documentation:
<https://github.com/pytorch/fairseq/blob/main/examples/roberta/README.md>
 - Download URL: <https://dl.fbaipublicfiles.com/fairseq/models/roberta.large.mnli.tar.gz>

Transfer Learning



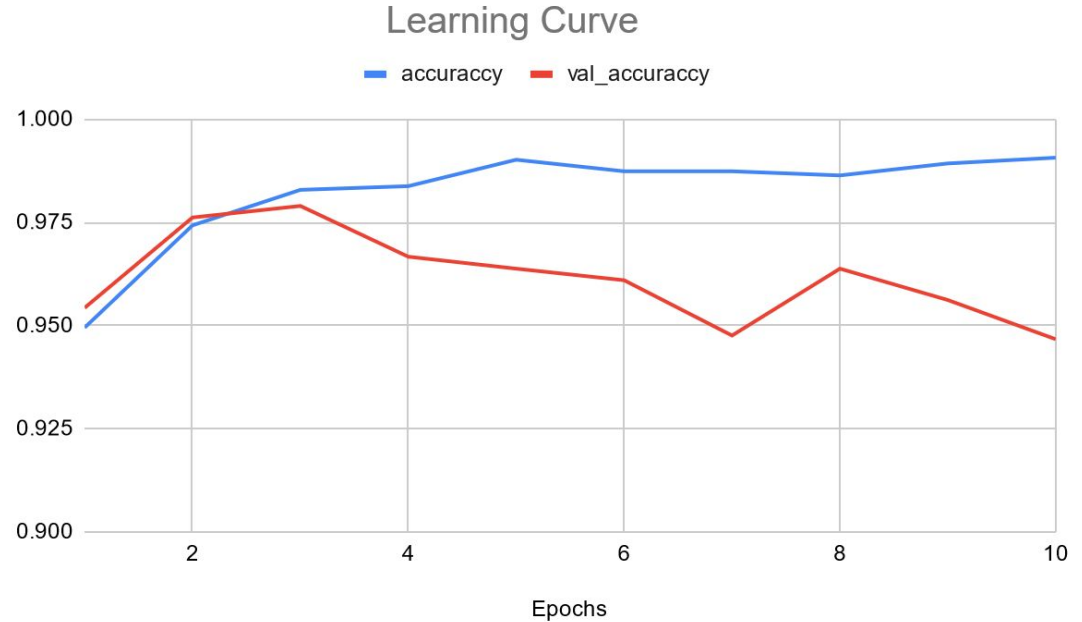
- Pretrained Weights [7]
 - Weights from the two pre-trained models below were loaded and further trained on the given data.
 - The models were trained fully with no frozen layers.
- English Model Layers
 - Input: shape has a max length = 120
 - Encoder = **roberta-large-mnli**
 - Pooling = GlobalAveragePooling1D
 - Output: 3 units, softmax , L1 regularization
- Multi Language Model
 - Input = shape has a max length = 120
 - Encoder = **xlm-roberta-large-xnli**
 - Pooling = GlobalAveragePooling1D
 - Output: 3 units, softmax , L1 regularization

Our Model Parameters

- L1 regularization with a penalty = .01 (default value)
 - L1 = Lasso Regression. Calculates the absolute value of the coefficients which means some features can potentially be eliminated entirely [8]
- Sparse Categorical Cross Entropy Loss [6]
 - We use this loss function when there are two or more label classes
 - Targets are represented by the index of the category
 - More efficient than Categorical_Cross_Entropy for NLP due to many categories
- Adam Optimizer [5]
 - adaptively estimates first and second moment (explain moments)
 - Separate step size for each derivative of the gradient
 - Learning rate adapts itself
- Learning Rate Schedule [9]
 - Customized defined learning rate that changes over the batches
 - Appropriate for Adam?
 - “The marginal value of adaptive gradient methods in machine learning”
 - The article summarizes that a customized learning schedule can generalize better than the built in adaptive method
 - Exponential Decay - decreases faster at the beginning and then flattens out later
 - Starts with 2×10^{-6} Learning Rate
 - Learning rate decays by 0.9 every 1000 batches

Learning Curve

Epoch	accuracy	val_accuracy
1	0.9495	0.9543
2	0.9743	0.9762
3	0.9829	0.979
4	0.9838	0.9667
5	0.9902	0.9638
6	0.9874	0.961
7	0.9874	0.9476
8	0.9864	0.9638
9	0.9893	0.9562
10	0.9907	0.9467



3 epochs was most optimal

Team Performance

LB URL: <https://www.kaggle.com/c/contradictory-my-dear-watson/leaderboard>

Metric: Accuracy: the percentage of correct prediction.

Formula: $\text{Accuracy} = (\text{all correct}) / \text{all} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$

As of:	10/18	10/25	11/1	11/8	11/15	11/22	11/29	12/6	Final Score
LB score	0.64600	0.65158	0.70202	0.93070	0.93089	0.93493	0.93705	0.94051	0.94051
LB rank	38	32	32	11	9	8	6	8	8
Entries	3	2	3	4	4	3	2	3	3

Knowing the true rank was difficult as some of the top models seemed to be periodically removed by Kaggle Admin. At the time of our Final Score, the top two contestants had accuracy values above 99%.

Modeling Challenges Below 90%

1. The Kaggle Competition recommended a TPU, but we found that for embeddings and transformer architectures to be very slow. Using a GPU improved runtime significantly.
2. Using a single multi language model was successfully deployed with accuracy up to 93%! Unfortunately, this caused the model to become infeasible due to very long train time.
3. Two vectors were run through our Neural Network: a separate embedding for the premise and hypothesis using SBERT. This actually made things worse.
4. Utilizing two models (English and non-English) within a single collab session is not possible due to memory constraints. Weights must be saved outside of collab and reloaded for each session. The weight files are large (each 3-4GB)

Modeling Challenges Below 90%

5. Overfitting: a high training accuracy of around 98% was achieved within 2 epochs, when our test accuracy was around 90%
6. We found that dropout (with a rate of 0.1) did not improve the model.
 - i. → Other nodes are not compensating for each other
7. We tried Nadam Optimizer and Adam actually did better
 - i. →Nadam Optimizer is just Adam with Nesterov momentum, also called Nesterov Accelerated Gradient/NAG [4]
 - ii. Momentum is added to both current gradient position and gradient change
 - iii. Result is gradient is based on new position rather than current position

Modeling Challenges Above 90%

Once accuracy rose above 90%, we encountered unique challenges:

1. Aurélien Géron in “Hands-On Machine Learning” recommend training our output layer weights with a high learning rate using the weights from the frozen pretrained model. Then with a much smaller learning rate, training all the layers together. Unfortunately, freezing several different numbers of layers all did not improve accuracy
2. Improving the Multi-language model was much more difficult than improving the English only model
 - a. Multi-language had a smaller dataset
3. Our validation set failed to have a meaningful effect:
 - a. Validation set was 20% of our data which further exacerbated the under-representation of some languages in the Multi-Language Set
 - b. The tuning on just 20% of the data failed to capture how to fix the few misclassified examples

Modeling Challenges Above 90%

4. Several strategies that normally increased accuracy failed to have effect
 - a. Additional hyper-parameter tuning and changing the number of top layers
 - b. GlobalAverage Pooling was more effective than both max and average pooling
 - c. Increasing the epochs required a with a tradeoff in learning rate
 - d. We tests several epochs ranging from 1-10
 - e. Too many epoch resulted in overfitting, too low resulted in low accuracy, we settled around 4 epochs

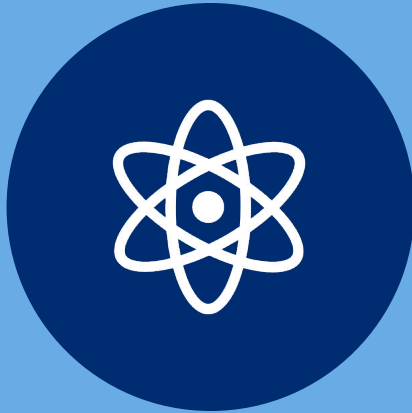
5. We reached an ostensible cap in accuracy at 93%, so we tried dramatically different architectures:
 - a. We tested numerous additional pretrained models for transfer learning in order to find the best model for our data.
 - b. We tested on 1 model instead of our usual two models and were able to get to a matching 93%

Discussion

If we had more time, we would consider several more directions:

- **Custom Model:** Of course, while the pretrained models were useful, a custom model specific to this dataset would likely produce much higher accuracy values
- **Segmentation:** We believed the a major source of error was not making classifications specific enough to misclassified examples. Therefore, we planned to first segment our data and then run a ensemble of models for each segment
 - Segment our dataset for each language or collection of similar languages
 - Segment our dataset using decisions trees
 - Segment our dataset using clustering
 - Identifying and then filtering our phrases or words in a sentence that were least likely to cause overfitting and entailment
- **Mitigate Overfitting:** On the contrary, a larger problematic theme was insufficient generalization
 - We needed to to capture more patterns between words in a sentence.
 - Longer sentence sizes → captured by larger batch size → more memory
 - Our current batch size was at 16 which meant our model would generalize patterns based on 16 weights, if we could increase the batch size, then we could capture more patterns
 - One of our competitors was processing full sentences
 - We estimated that this would require about 64 GB of GPU memory with 32 capturing most sentences.
 - Our limitation was we were running about 14.88 GB of GPU for half sentences with Colab Pro Plus

Q & A



We are happy to answer your questions! Please email us:

Sina Azartash	<u>sazarta1@jh.edu</u>
Jeff Simon	<u>jsimon34@jhu.edu</u>
Linshen Wu	<u>lwu53@jhu.edu</u>
Yaron Walfish	<u>ywalfis1@jhu.edu</u>

References

1. *Entailment* | *Linguistics* | *Glossary* | *Ultius*. (2021). Entailment. <https://www.ultius.com/glossary/linguistics/entailment.html>
2. Briggs, J. (2021, September 2). *Train New BERT Model on Any Language* | *Towards Data Science*. Medium.
<https://towardsdatascience.com/how-to-train-a-bert-model-from-scratch-72cfce554fc6>
2. NoteBooks Grandmaster, Mr_KnowNothing. (2020, April 13). *Deep Learning For NLP: Zero To Transformers & BERT*.
Kaggle.
<https://www.kaggle.com/tanulsingh077/deep-learning-for-nlp-zero-to-transformers-bert#BERT-and-Its-Implementation-on-this-Competition>
3. Schmidt, D. (2018, November 23). *Understanding Nesterov Momentum (NAG)*. Dominik Schmidt.
<https://dominikschmidt.xyz/nesterov-momentum/>

References

5. Brownlee, J. (2021, October 11). *Gradient Descent Optimization With Nadam From Scratch*. Machine Learning Mastery.
<https://machinelearningmastery.com/gradient-descent-optimization-with-nadam-from-scratch/>
6. *tf.keras.losses.SparseCategoricalCrossentropy* | *TensorFlow Core v2.7.0*. (2021). TensorFlow.
https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy
7. Gupta, D. (2021, January 4). *Transfer Learning | Pretrained Models in Deep Learning*. Analytics Vidhya.
<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/>
8. S. (2020, July 7). *Regularization: Simple Definition, L1 & L2 Penalties*. Statistics How To.
<https://www.statisticshowto.com/regularization/>
9. Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. 2017. The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4151–4161.

Deep Neural Networks: EN.605.742.81.FA21



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING

Jeff Simon, Yaron Walfish, Sina Azartash, Linshen Wu

Final Project