GLUE Benchmark:

Final Project Report

NLP for Data Science

DATS 6312

Arathi Nair, Stefani Guevara, Mariko McDougall

12-09-2021

Table of Contents

1. Introduction

The General Language Understanding Evaluation benchmark (GLUE) is a compilation of natural language datasets and tasks designed with the goal of testing models on a variety of different language challenges.

Historically, many NLP models were trained and tested on very specific datasets. This method often produced well performing models when run on similar data, but had very poor performance when applied to corpuses that varied from their original training data. Additionally, these models could only be used to perform the specific task they were built for, with very little cross-utility to different tasks, even when using the same data.

With the advent of transfer learning, it became possible to create a model which could be used for multiple tasks, across a variety of input datasets. Through this, models that have a deeper understanding of language can be created, with broad cross-utility and applicability in a variety of natural language contexts.

The 9 tasks in the GLUE dataset represent a diverse set of challenges, from grammatical parsing to context-reliant pronoun identification, from an array of text contexts, including news reports, public forums and wikipedia pages.

To create a high-performing model capable of understanding and generating natural language, we integrate three state-of-the-art models together with an ensemble learning model to capture information from the three disparate models on each task. With this method, we capitalize on the unique strengths of each individual model, while mitigating most weaknesses.

2. Description of the Datasets

The GLUE dataset consists of 9 tasks, which can be categorized into 3 broad groups: Single sentence tasks, Similarity and Paraphrase tasks, and Inference tasks. Each of the tasks have discrete datasets, but within groups the datasets may be similar. Below we discuss the individual datasets by task category.

Single Sentence Tasks

The Corpus of Linguistic Acceptability (CoLA)

The CoLA dataset consists of pairs of English sentences along with labels indicating if the sentence is in correct grammatical form.

The Stanford Sentiment Treebank (SST-2)

The SST-2 dataset consists of sentence excerpts from human-written movie reviews and labels indicating positive or negative sentiment. The labels for this data are continuous from 0 to 1, with 0 indicating negative sentiment.

Similarity and Paraphrase Tasks

Microsoft Research Paraphrase Corpus (MRPC)

The MRPC dataset consists of pairs of sentences taken from online news articles and labels indicating if the sentence pairs are semantically equivalent. Notably, the classes are imbalanced in this dataset.

The Quora Question Pairs (QQP)

The QQP dataset is similar to the MRPC dataset in that it also consists of pairs of sentences, this time taken from Quora, an online question-response website, and labels indicating whether the sentence pairs are semantically equivalent. QQP also contains class imbalance, as in MRPC.

The Semantic Textual Similarity Benchmark (STS-B)

The STS-B dataset also consists of sentence pairs, this time drawn from headlines, image and video captions and other data. Here, the labels indicate the similarity between the sentences as a score between 1 and 5, as opposed to whether they are equivalent (0 or 1) as in the MRPC task.

Inference Tasks

The Multi-Genre Natural Language Inference Corpus (MNLI)

The MNLI dataset consists of sentence pairs, with a hypothesis sentence followed by a premise sentence sourced from transcribed speech, fiction and government reports. The labels for the data indicate if the premise sentence supports (*entailment*), refutes (*contradiction*) or neither supports nor refutes the hypothesis (*neither*). Of note, the data consists of both matched in-domain hypothesis-premise pairs, and mismatched cross-domain pairs.

The Stanford Question Answering Dataset (QNLI)

The QNLI dataset consists of sentence pairs, with the first sentence being a question, and the second containing text from wikipedia which potentially answers the question posed. The labels for the dataset indicate if the answer to the question is present in the second sentence.

The Recognizing Textual Entailment (RTE)

The RTE dataset contains sentence-pair entailment data, similar to the MNLI dataset. It combines several years of entailment challenge datasets (RTE1, RE2, RTE3 and RTE5) into one large corpus, with text drawn from Wikipedia and news articles. As with the MNLI dataset, the pairs consist of a hypothesis sentence and a premise sentence which may or may not support (entail) the hypothesis. Unlike MNLI, there are only two classes in RTE, *entailment* and *not_entailment*.

The Winograd Schema Challenge (WNLI)

The WNLI dataset consists of sentence pairs, with the first sentence containing an ambiguous pronoun that must be inferred by the context of the sentence, and the second sentence containing a proposed referent in the context of the sentence. The labels for the dataset indicate if the proposed pronoun is entailed by the original sentence, or if it is inappropriate. The classes for this dataset are imbalanced.

Corpus	Task	Metric	Domain
Single-Sentence Tasks			
CoLA (Corpus of Linguistic Acceptability)	*Grammatical Correctness*	*Matthew's correlation coefficient*	*Linguistics literature*
SST-2 (Stanford Sentiment Treebank)	*Sentiment Analysis*	*Accuracy*	*Movie reviews*
Similarity and Paraphrase Tasks			
MRPC (Microsoft Research Paraphrase Corpus)	*Paraphrase detection of two sentences*	*Accuracy and F1 Score*	*News*
QQP (Quora Question Pairs)	*Paraphrase detection of two questions*	*Accuracy and F1 Score*	*Social QA Questions*
STS-B (Semantic Textual Similarity Benchmark)	*Sentence similarity*	*Pearson and Spearman correlation*	*Misc.*
Inference Tasks			
MNLI (Multi-Genre Natural Language Inference Corpus)	*Sentences match/mismatch*	*Accuracy*	*Misc.*

QNLI (Stanford Question Answering Dataset)	*Question & Answer pairing*	*Accuracy*	*Wikipedia*
RTE (Recognizing Textual Entailment)	*Sentences match/mismatch*	*Accuracy*	*News & Wikipedia*
WNLI (Winograd Schema Challenge)	*Sentences match/mismatch with pronoun substitution*	*Accuracy*	*Fiction books*

Table 1: A breakdown of the objective, evaluation metric, and source of dataset for each GLUE task.

3. Description of the NLP Model and Algorithm

To develop a dynamic and well-rounded overall model, we leveraged the power of three independent state-of-the-art natural language processing transformers models. To maximize the information utilization, the outputs of the three independent models were combined together with a weighted random forest classifier. Below, we outline each model, and our method of combining them.

ELECTRA

ELECTRA is a method for self-supervised language representation learning. The ELECTRA model's architecture has a unique pre-training approach called 'replaced token detection' unlike its predecessors that relied on MLM pre-training. The models are trained to distinguish "real" input tokens vs "fake" input tokens generated by another neural network, similar to the discriminator of a generative adversarial network (GAN). The model can efficiently learn and classify token replacements accurately by including all input tokens. The results obtained by the model can match or exceed the pretrained MLM model using relatively little computation. At a small scale, ELECTRA achieves strong results even when trained on a single GPU.

The ELECTRA framework consists of two networks - a generator and a discriminator; wherein these two networks are pit against each other. The generator is optimized to trick the discriminator with increasingly convincing "fake" data. The discriminator is then required to identify and separate the true inputs from the fake data. However, the generator of the ELECTRA model is not optimized to increase the loss of the discriminator but is instead trained like a typical MLM model where it must best guess the "[MASK]" values. This approach to pre-training is more efficient than MLM because the model must consider every single token. This produces a model which has an improved comprehension of context.

After pre-training is complete, the generator model is discarded, and the new ELECTRA transformer model can then be applied to handle various language processing tasks.

Unlike GAN, the generator is trained using maximum likelihood rather than being trained to fool the discriminator. The generator outputs the probability of generating the token at a particular position using the Softmax layer and the discriminator predicts whether the token at that position is “real” or “replaced” using Sigmoid activation function. Finally, the combined loss of generator and discriminator is minimized before discarding the generator.

XLNet

XLNet is a generalized autoregressive model developed in 2019 by researchers at Carnegie Mellon and Google AI. There were two main issues XLNet intended to solve of the then-championed SOTA BERT model: overcoming the model’s independence assumption and its data corruption required pre-training method. On the former, BERT assumes that all predicted/masked tokens are dependent on all input tokens, but independent of each other. However, this is not the case in natural language where there are high-order, long-range dependencies. On the latter, BERT mandates data corruption in pre-training through inserting artificial symbols such as [MASK] on target tokens, which causes a discrepancy with actual data during fine-tuning as these special tokens are not included in real data.

To address these issues, XLNet uses the relative positional embedding and recurrence mechanism of the Transformer-XL architecture, a permutation operation, and two-stream self-attention. The architecture of the Transformer-XL model allows XLNet to propagate positional information of previous segments thereby applying a general autoregressive method where the model uses previous inputs to learn dependencies in a feedforward fashion, without the use of delays as in recurrent neural networks. While retaining importance and relative position of previous segments, thus capturing long-term dependencies, the permutation operation with XLNet allows the model to capture bi-directional context in the current segment by maximizing its log-likelihood w.r.t. all possible permutations. More concretely, the model learns to gather information from all positions because it makes predictions not from left-to-right but by randomly sampling parts of the current segment and making predictions on those samples, thus learning dependencies between all combinations of inputs. Both help the model learn token dependencies better than BERT.

However, since the model uses permutation to learn bidirectional context, it also needs to know the position of the target token to make valid predictions. Thus, the model incorporates position awareness through its two-stream, query and content, attention mechanism. The query stream maintains the positional encoding but does not have access to the target token while the content stream holds the encoded permutation (context information) and thus can perform normal self-attention. The query stream uses the information from the content stream to predict the target token.

While XLNet leverages autoregressive language modeling with bidirectional learning, these come at a heavy cost. The model is notably computationally expensive and takes much longer to run than its counterparts in this experiment. The permutation operation across input sequences, the core of XLNet’s strategy, requires additional resources that were limited to us during this project. To avoid overextending our GPU, we had to maintain the batch size within the 5 to 10 range, depending on the

task, and applied gradient accumulation through the eval_accumulation_steps parameter in our TrainingArguments method.

DeBERTa

DeBERTa is a BERT-based model produced by Microsoft Dynamics and Microsoft Research in October 2021. DeBERTa is an abbreviation of **D**ecoding-**E**nhanced BERT with disentangled **A**ttention, and as the name implies there are two major innovations in this model compared to previous similar innovations: an Enhanced Decoding mask and Disentangled attention. As we will cover more thoroughly, both of these innovations stem from an enhanced tracking of the position of words within a given sentence.

As a BERT-based model, DeBERTa is trained using masked-language-modeling (MLM), where ingested text has words randomly masked, and the model attempts to predict the original words. While the base BERT model tracks position of words within a sentence using a position embedding, this embedding is then combined with word embedding. The attention mask is then calculated using the combined position and content embeddings. However, this process does not allow for using the position of the word as a feature in and of itself. DeBERTa departs from this method by keeping contents and position as separate matrices, and calculates the attention masks using both separate features. By disentangling position and content, it is possible to make more informed inferences about the context of words in a sentence.

To further take advantage of the enhanced tracking of positionality, DeBERTa incorporates absolute position of words in the sentence, in addition to the relative position captured using the disentangled attention. This allows even more fine-tuned comprehension of syntax, given position in the sentence. For example, in the sentence “a new store opened beside the new mall”, despite similar contexts and meanings for the words “store” and “mall”, due to the absolute positions of the words we can determine that “store” is the subject of the sentence. To capture the absolute position, DeBERTa creates an embedding layer of the absolute position of each word, and incorporates it before the final softmax layer where the model decodes masked words. This is the “Enhanced Mask Decoder” aspect of the model.

Ensemble Learning

To integrate the predictions from all models and select the most probable answer, we used ensemble learning. Ensemble learning is a meta approach to modeling that leverages the power of multiple predictive models to make predictions that are more accurate than any model in the ensemble. Whereas many approaches would compare the outputs of each model and simply take the best result, ensemble learning takes the class predictions from all models and uses all of them as an input. This integration, rather than exclusion of less accurate models allows the ensemble learner to glean additional information from the weaker models as well, rather than discarding it.

This method of combining multiple models frequently produces more accurate results than any individual model in the ensemble.

In this project, we use both Random Forests and Gradient Boosting ensemble methods, and compare and contrast their results. Gradient boosting and random forests are very similar models, both are decision tree-based and mainly vary with how the output of the trees are combined. Where random forests build all trees simultaneously and independently (horizontally), gradient boosting builds them sequentially, with each tree being built in a chain. Both methods have their own strengths and weaknesses, and so we thought to run both, and compare the outputs.

With each ensemble method we use both the classifier and regressor variants. This is necessary due to the nature of our datasets - while many have class based or binary predictions which can be predicted with a classifier model, STS-B has a continuous variable which measures the similarity of the two sentences. This continuous variable must be obtained with a regressor, and cannot be obtained with a classifier.

4. Experimental setup

Data

The GLUE dataset is a composite dataset consisting of 9 distinct tasks. To easily load and process each of these tasks, we employ the ‘datasets’ package provided by Hugging Face, thus the GLUE tasks are conveniently accessible using the load_dataset() function. The loaded datasets are pre-separated into training, testing and validation datasets. As each is also associated with a specific evaluation metric, we also use the load_metric() function from the datasets package to import the task’s respective evaluation metric.

Implementation of Framework

The overall framework of our model is to train each of our transformer models on each task independently, then combine the predictions of each trained model. These combined predictions are then used as an input for an ensemble model.

We will test both Random Forests and Gradient Boost ensemble methods, and compare their outputs against each other and against the individual transformer models. For both ensemble methods, we use a regressor model (RandomForestRegressor() and GradientBoostingRegressor()) to predict the sts-b task, the result of which is a continuous variable. For all other tasks we employ a classifier model (RandomForestClassifier() and GradientBoostingClassifier()) to predict discrete classes. All ensemble methods are imported from sklearn.ensemble. For both ensemble techniques we perform a fit using the default settings for the regressor ensemble method as well as a hypertuned version using GridSearchCV for the classifier ensemble methods.

Baseline Comparison

To establish a baseline of performance, each model was initially evaluated on each task of the GLUE dataset without first training on the task's training dataset. Transformer models perform best when trained on a specific dataset, like most models, but are nonetheless capable of performing adequately without training. This untrained model acts as a baseline for the individual model's trained performance on each task.

A secondary baseline is then established for the ensemble model. When creating the predictions on which to train the ensemble model, we also evaluate the model on the same dataset. These accuracy scores can then be compared between each individual model and the ensemble model.

The scores for each GLUE task is then compared for each trained transformer model, as well as for each ensemble model.

5. Hyperparameters

Hyperparameters

ELECTRA

After obtaining a baseline score for all the GLUE tasks, the model was fine tuned to achieve higher prediction scores and improve the performance of the model. For hyperparameter tuning, we experimented with different learning rates (5e-5, 2e-5, 1e-4), weight decay values (0.01and 0.1), number of epochs (3, 5,10) and batch sizes (8,10,12). The optimal results were obtained by setting the learning rate = 2 e-5, weight decay = 0.01, number of epochs = 3, batch size = 8 (except for WNLI task which was set as 10).

To further fine-tune the model parameters, we also implemented hyperparameter search using Optuna which is an automated optimization software that uses different machine learning algorithms such as grid search, random, Bayesian. However, the results achieved using Optuna did not yield higher scores compared to the previously fine-tuned model.

XLNet

Hyperparameter tuning was extremely limited with XLNet since, as noted in section 3, the model, even the base-cased version used in this project, is resource intensive. When attempting to finetune on a given task, the model would crash early on or as much as 2 hours into training. Especially when hyperparameter tuning with Optuna, when training only 5 trials, most tasks would take hours to run and often crash prior to completing. It did not seem worth pursuing hyperparameter tuning extensively since, as discussed above with Electra, the results that were obtained with Optuna on certain tasks were not competitive to the already achieved scores.

DeBERTa

As with XLNet, DeBERTa received limited hyperparameter tuning due to limitations in GPU capacity and maximum training duration. This is partially attributable to the large size of the model, as despite using the smallest current version of DeBERTA, ‘deberta-v3-small’, the model nonetheless occupied a large portion of the GPU on its own. Further, as with both Electra and XLNet, the results obtained with Optuna were not significantly better than the trained but untuned models.

Ensembles

Hyperparameter tuning for the Random Forest ensemble involved tuning via GridSearchCV on the number of trees in the forest, max number of features to consider in a split, and the minimum number of samples per node. GridSearchCV was also used to hypertune the Gradient Boosting ensemble, tuning on the type of loss (deviance or exponential), number of boosting stages, and the max number of features to consider in a split.

Overfit Detection

Generally speaking, transformer models are very resilient to overfitting, given their extremely large training set prior to distribution. Due to their natural language understanding, they are less likely than a standard model to dramatically overfit in general. For the GLUE tasks in specific, the size of the training set is sufficiently small that overfitting a transformer model using only the training set is unlikely.

6. Results

ELECTRA

ELECTRA			
Corpus	**Metric**	**Baseline**	**Tuned**
Single-Sentence Tasks			
CoLA	*Matthew’s correlation*	0.466	0.607
SST-2	*Accuracy*	0.883	0.917

Similarity and Paraphrase Tasks			
MRPC	*Accuracy and F1*	'accuracy': 0.742, 'f1':0.805	'accuracy': 0.865, 'f1': 0.903
QQP	*Accuracy and F1*	'accuracy': 0.879, 'f1': 0.839	'accuracy': 0.898, 'f1': 0.863
STS-B	*Pearson and Spearman*	'pearson': 0.548, 'Spearmanr': 0.524	'pearson': 0.877, 'spearmanr': 0.875
Inference Tasks			
MNLI	*Accuracy*	0.799	0.820
MNLI-MM	*Accuracy*	0.825	0.821
QNLI	*Accuracy*	0.866	0.889
RTE	*Accuracy*	0.555	0.682
WNLI	*Accuracy*	0.436	0.603

From the above results, we observe that fine tuning the trained baseline model drastically improved the performance. The ELECTRA-Small model was able to obtain a high accuracy in classifying the labels and predicting the output for most GLUE benchmark tasks. Overall, we find that the model is most likely to perform higher on textual similarity, rather than on an actual understanding of the text.

XLNet

XLNet			
Corpus	**Metric**	**Baseline**	**Tuned**
Single-Sentence Tasks			
CoLA	*Matthew's correlation*	0.000	0.399
SST-2	*Accuracy*	0.505	0.940

Similarity and Paraphrase Tasks			
MRPC	*Accuracy and F1*	0.686 f1: 0.812	0.892 f1: 0.922
QQP	*Accuracy and F1*	0.372 f1: 0.539	0.874 f1: 0.893
STS-B	*Pearson and Spearman*	P: 0.021 S: 0.010	P: 0.893 S: 0.889
Inference Tasks			
MNLI	*Accuracy*	0.314	0.857
MNLI-MM	*Accuracy*	0.312	0.858
QNLI	*Accuracy*	0.495	0.878
RTE	*Accuracy*	0.459	0.740
WNLI	*Accuracy*	0.578	0.563

The table above shows that while XLNet was able to generate predictions on the baseline -- even fair predictions on the MRPC dataset -- it had a stark improvement when predicting on the validation set with basic fine-tuning. This applies to all datasets save for that of WNLI in which the baseline shows marginally better performance than the fine-tuned model, just as in the case with DeBERTa below. We can also see from the table that XLNet did particularly well on similarity and paraphrase tasks and remained competitive with its counterparts in this project in datasets within each task category.

DeBERTa

DeBERTa			
Corpus	**Metric**	**Baseline**	**Tuned**
Single-Sentence Tasks			
CoLA	*Matthew's correlation*	0.000	0.621
SST-2	*Accuracy*	0.491	0.935

Similarity and Paraphrase Tasks			
MRPC	*Accuracy and F1*	0.316 f1: 0.000,	0.865 f1: 0.903
QQP	*Accuracy and F1*	0.632 f1: 0.000	0.906 f1: 0.875
STS-B	*Pearson and Spearman*	P:0.055 S:0.067	P:0.868 S:0.868
Inference Tasks			
MNLI	*Accuracy*	0.354	0.875
MNLI-MM	*Accuracy*	0.330	0.872
QNLI	*Accuracy*	0.495	0.915
RTE	*Accuracy*	0.527	0.668
WNLI	*Accuracy*	0.563	0.437

In nearly all tasks, there was a strong improvement over baseline when the DeBERTa was trained on the tasks prior to evaluating on the validation set. Interestingly, this is not true for all tasks; WNLI performs slightly better on the baseline untrained model compared to the trained model. This is an interesting quirk of the WNLI dataset, which is known to be mildly adversarial: several training examples have very similar, but opposite, examples in the testing and validation set. This often leads to lower scores on the WNLI dataset, to the point where it is specifically addressed in the GLUE dataset's FAQ.

Ensemble Learning

		Transformers			**Ensemble Methods**	
Corpus	**Metric**	**Electra**	**XLNet**	**Deberta**	**Random Forest**	**Grad Boost**
Single-Sentence Tasks						
CoLA	*Matthew's correlation*	0.607	0.400	0.621	**0.688**	0.657

SST-2	*Accuracy*	0.917	0.940	0.935	**0.958**	0.954
Similarity and Paraphrase Tasks						
MRPC	*Accuracy and F1*	0.882 f1:0.915	**0.892** **f1:0.923**	0.865 f1:0.903	0.878 f1:0.912	0.854 f1:0.893
QQP	*Accuracy and F1*	0.900 f1:0.866	0.874 f1:0.833	0.906 f1:0.875	0.909 f1:0.878	**0.910** **f1:0.879**
STS-B	*Pearson and Spearman*	P:0.873 S:0.872	P:0.893 S:0.889	P:0.868 S:0.868	P:0.894 S:0.889 *Note: Regressor	**P:0.897** **S:0.892** ***Note: Regressor**
Inference Tasks						
MNLI	*Accuracy*	0.817	0.857	0.875	**0.877**	**0.877**
MNLI-MM	*Accuracy*	0.821	0.858	0.872	0.875	**0.877**
QNLI	*Accuracy*	0.889	0.878	0.915	0.916	**0.924**
RTE	*Accuracy*	0.682	**0.740**	0.668	0.726	0.702
WNLI	*Accuracy*	0.465	0.563	0.437	0.500	**0.636**

The ensemble methods overall outperformed the individual Transformer models, with the exception, interestingly, of XLNet in the MRPC and RTE tasks. Pit against one another the Random Forest and Gradient Boosting ensembles remained competitive, at times one outperforming the other by a margin of 0.2%. While Random Forest tended to do better on the single sentence tasks, Gradient Boosting generally did better on the similarity and inference tasks.

7. Summary and Conclusions

Results

When pitting Transformer-to-Transformer, we found XLNet and DeBERTa performed competitively against each other, even with the minimal hyperparameter tuning on each. Out of the 9 tasks plus the MNLI-MM, DeBERTA and XLNet outperformed the other on 5 tasks each. XLNet is not as strongly suited for short sequences given its pretraining on long-range dependencies and masking, in its own way, during permutation; we find this to be a likely reason why the model was beaten in the QQP and QNLI tasks. Questions, inherently, are short sequences where sentences can be much longer.

While our Gradient Boosting Ensemble did outperform the other models included in this project on several tasks, it did not beat the Random Forest Ensemble in the Single-Sentence Tasks. Moreover, and surprisingly, both Ensembles were beaten by XLNet on two tasks, MRPC and RTE, likely due to XLNet's strength of capturing long-term dependencies.

Learning

We found that ensembling works well and expectedly outperforms stand-alone Transformer models when used for text classification. We also found that for the most part, the Gradient Boosting ensemble did the best on the GLUE tasks, but we were surprised that the base XLNet version was able to beat either ensemble in a couple of tasks. It would be interesting to see how the XLNet-Large would do, if we had the resources to train on it.

Improvements

More extensive fine-tuning would likely improve model scores. However, to perform this fine-tuning more permissive ec2 security protocols would be required, as the security protocol requires access through GWU IP addresses. As no group members lived on campus, we used a VPN which unfortunately had a time-out after 12 hours. This time limit was surprisingly limiting for our training process, and necessitated that all training intervals complete in less than 12 hours lest the connection cut-off.

Access to more GPUs or to TPUs would also help improve scores as models such as XLNet are more computationally expensive but can produce better results with more RAM.

Additional models may increase the ensemble performance, as long as they are distinct from the models used in the current ensemble. Another BERT-like model, for instance, may not increase performance by a significant margin, due to its similarity to DeBERTa.

8. References

In addition to references used for background information or for the written portion, you should provide the links to the websites or github repos you borrowed code from.

GLUE Explained: Understanding BERT Through Benchmarks
Dataset description sheet
GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTANDING
DeBERTa Original Paper
Hugging Face Transformers Examples
ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators.
Paper Reading #2: XLNet Explained
XLNet: Generalized Autoregressive Pretraining for Language Understanding
Guide to XLNet for Language Understanding
Data Science Central
XLNet Explained
XLNet: Generalized Autoregressive Pre-training for Language Understanding