Improving Logical Consistency in Pre-Trained Language Models using Natural Language Inference

Stanford CS224N Custom Project

Ananth Agarwal
Department of Computer Science
Stanford University
ananthag@stanford.edu

Cameron Tew
Department of Computer Science
Stanford University
ctew@stanford.edu

Anthony Tzen
Department of Computer Science
Stanford University
atzen@stanford.edu

Abstract

Current state-of-the-art pre-trained language models (PTLMs) contain rich and vast amounts of world knowledge, demonstrating an ability to extrapolate information from contextual texts and to accurately answer questions [1]. However, the latent factual understanding captured by PTLMs can be irrational and inconsistent, causing PTLMs to be prone to generating contradictory statements [2]. We demonstrate that natural language inference (NLI) can provide additional signal about contradictory statements output by a PTLM. We explore several approaches for aggregating the entailment and contradiction probabilities acquired through NLI on a batch of PTLM predicted answers and define a scoring heuristic that balances between the NLI output and the PTLM's confidence in its answers. Predictions whose scores are below a tuned threshold are revised before outputting final answers. In addition, we investigate methods for using these NLI probabilities to define a MaxSAT problem that, when optimized, yields corrected predictions. Our results demonstrate that a system that uses either of our approaches to revise PTLM answers has better accuracy and logical consistency than a vanilla PTLM.

1 Key Information to include

- Mentor: Eric Mitchell (em7@stanford.edu)
- External Collaborators (if you have any): n/a
- Sharing project: n/a

2 Introduction

Pre-trained language models (PTLMs) are deep transformer-based neural networks trained over large corpora of text. PTLMs can be easily distributed and fine-tuned to many different tasks across NLP. Consistency in generated answers for question answering tasks is a property that would indicate a PTLM has learned a robust set of beliefs. Modern state-of-the-art models, however, generate surprising logical inconsistencies, indicating they are falling short of acquiring human-like factual understanding.

For example Macaw [3], a PTLM built on T5, outputs the following logical contradiction for a simple question answering task:

- Q: Is an american bison a mammal? A: Yes
- Q: Is a mammal a bird? A: No
- Q: Is an american bison a bird? A: Yes

The prevalence of logical inconsistencies like the one above demonstrates the need to improve how PTLMs reason about the world and the knowledge they have captured.

Natural language inference (NLI) models are trained on the sequence classification task of determining whether the relation between a premise and hypothesis is "entailment", "contradiction", or "neutral". For example a RoBERTA model [4] fine-tuned on Multi-Genre Natural Language Inference (MNLI) indicates that the affirmative statement "An american bison is a bird" generated by processing the Macaw answer above strongly contradicts with the premise "An american bison is a mammal". Our work focuses on using NLI to estimate contradiction and entailment probabilities between pairs of statements generated from PTLM question-answer results in order to determine whether the PTLM answer should be revised before outputting a final answer. We explore scoring heuristic methods using the data output from the NLI and the PTLM's confidence in its original answer. Our results demonstrate improvements over baseline PTLM accuracy and logical consistency in all cases.

3 Related Work

Li et al. show that highly accurate PTLMs often fail to exemplify consistency, for example in transitive logic [5]. Kassner et al. leverage a global persistent memory called BeliefBank and a constraint solver to build a system around a PTLM that improves accuracy and logical consistency [6]. The weighted MaxSAT constraint solver uses a manually curated set of logical implications as constraints to revise the PTLM's raw answers that most clash with other answers in the BeliefBank, and this processed answer is output. Limitations of Kassner et al.'s work include poor scalability of procuring manually curated constraints for a constraint solver, and that entities with multiple meanings are excluded from the dataset. The pre-trained NLI model we are utilizing in our approach has the advantages of internalized knowledge and contextualized representation of words.

4 Approach

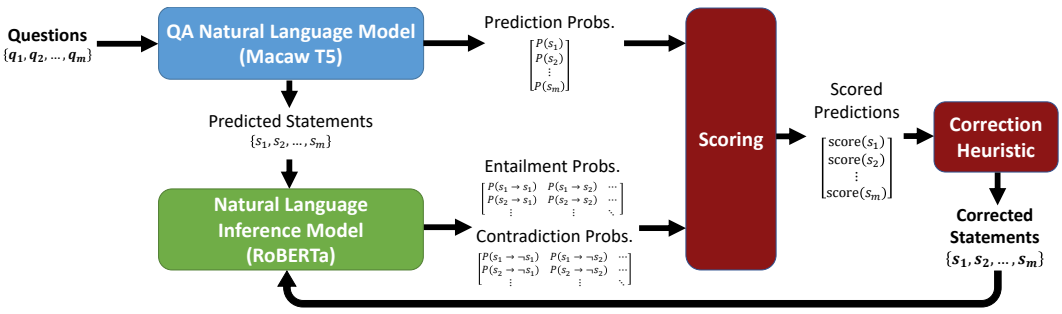

Figure 1: Scoring & Iterative Improvement Approach

4.1 Data Preprocessing

We first preprocessed the BeliefBank dataset curated by Kassner et al. in order to construct batches of factual statements for testing our model. The evaluation dataset contains facts that are either true or false about 85 entities ("silver facts" inferred from a knowledge graph; see Section 5.1), and it also contains a directed constraint graph that encodes entailment between pairs of statements [6]. We created one batch of test facts per entity in the following way:

1. Sample a statement about the entity from the set of silver facts and add it to the batch.
2. Using the constraints, sample and add a new statement that can be inferred from statements already added to the batch. If no such statements exist, go back to step 1.

3. Repeat step 2; stop when the batch has batch size n statements.

In addition, to ensure that each batch has a balanced sample of both positive and negative statements, we attempt to sample from only applicable positive statements if there are more negative statements in the current batch (and vice versa) if possible in steps 1 and 2. Overall this process ensures that for each batch, there are enough logical implication statement pairs to compute a reliable consistency metric, i.e., our model has the potential to make predictions about the batch's statements that would satisfy or violate a significant number of constraints. We can thus evaluate the logical consistency of a batch of statements by examining what percentage of these constraints are satisfied or not satisfied (Section 5.2).

4.2 Querying the PTLM and NLI

Our model takes one batch of facts at a time as input. Each fact is mapped to a natural English yes/no question using a simple template. For example our "IsA" relation template maps the raw input: `mammal:{IsA,bird:yes}` to "Is a mammal a bird?". The question is then fed into a QA PTLM which then outputs a yes/no answer along with its probability. Using templates again, we transform each question-answer pair into a declarative statement. For example <Q: "Is a mammal a bird?", A: "no"> is mapped to "A mammal is not a bird". The PTLM probability of its yes/no answer can subsequently be interpreted as the PTLM-estimated probability of the resulting statement; that is, for a question answer pair <q_i, a_i> and corresponding statement s_i, $P_{PTLM}(s_i) = P_{PTLM}(a_i)$.

We then evaluate the logical consistency of this batch of statements, $S = \{s_1, s_2, ..., s_n\}$. This is where we introduce a novel approach - rather than using a manually configured constraint solver like Kassner et al., we use a pre-trained NLI model, which takes a hypothesis statement s_h and a premise statement s_p as input and predicts an entailment probability, a contradiction probability, and a neutral probability. We denote the first two probabilities as follows:

$$P_e(s_h, s_p) = P_{NLI}(s_p \implies s_h)$$
$$P_c(s_h, s_p) = P_{NLI}(\neg(s_p \wedge s_h)) = P_{NLI}(s_p \implies \neg s_h))$$

Each ordered pair of statements (s_h, s_p) (where $s_h, s_p \in S$) is passed through the NLI to create two $n \times n$ matrices: entailment probabilities and contradiction probabilites.

4.3 Estimating $P(s_h)$ from Logical Relationships

The PTLM and NLI outputs are used to compute new estimates for the probability of each statement $s_h \in S$, using only the estimates of how the other statements in the batch entail or contradict with s_h. Specifically, for each (premise s_p, hypothesis s_h) pair of statements where $p \neq h$, we can estimate:

$$P(s_h) = P(s_p \wedge (s_p \implies s_h)) = P(s_p)P(s_p \implies s_h) = P_{PTLM}(s_p)P_e(s_h, s_p)$$
$$P(\neg s_h) = P(s_p \wedge (s_p \implies \neg s_h)) = P(s_p)P(s_p \implies \neg s_h) = P_{PTLM}(s_p)P_c(s_h, s_p)$$

Then, for each s_h, we aggregate the relevant estimates into a single $P_{NLI}(s_h)$ estimate, experimenting with several methods to do so. The first approach takes the **maximum** of the probability estimates. This closely matches the underlying logical reasoning: only one entailment clause $(s_p \wedge (s_p \implies s_h))$ needs to be true for s_h to be true, and only one contradiction clause $(s_p \wedge (s_p \implies \neg s_h))$ needs to be true for s_h to be false.

$$P_{NLI}(s_h) := P\Big(\bigvee_{p \neq h} s_p \wedge (s_p \implies s_h)\Big) = \max_{p \neq h} P(s_p \wedge (s_p \implies s_h) = \max_{p \neq h} P_{PTLM}(s_p)P_e(s_h, s_p)$$
$$P_{NLI}(\neg s_h) := P\Big(\bigvee_{p \neq h} s_p \wedge (s_p \implies \neg s_h)\Big) = \max_{p \neq h} P(s_p \wedge (s_p \implies \neg s_h) = \max_{p \neq h} P_{PTLM}(s_p)P_c(s_h, s_p) \tag{1}$$

Another method is to compute an empirical estimate by taking the **average** of the probability estimates. Here, a statement that is strongly entailed/contradicted by more statements can have a higher probability estimate than a statement with just one equally strong or stronger entailment/contradiction.

$$P_{NLI}(s_h) := \frac{1}{|S|-1} \cdot \sum_{p \neq h} P(s_p \wedge (s_p \implies s_h)) = \frac{1}{|S|-1} \cdot \sum_{p \neq h} P_{PTLM}(s_p)P_e(s_h, s_p))$$
$$P_{NLI}(\neg s_h) := \frac{1}{|S|-1} \cdot \sum_{p \neq h} P(s_p \wedge (s_p \implies \neg s_h)) = \frac{1}{|S|-1} \cdot \sum_{p \neq h} P_{PTLM}(s_p)P_c(s_h, s_p)) \tag{2}$$

A third method is to compute an average of the entailment and contradiction probabilities weighed by the PTLM estimates of the premise probabilities. This **weighted average** approach is similar to the previous approach, but the entailment and contradiction probabilities are scaled by the *relative* values (instead of the absolute values) of the premise probabilities. We can interpret these scaled PTLM premise probabilities as a probability distribution $P_{scaled} = P_{PTLM}(s_p|S\backslash\{s_h\})$, where each premise's scaled probability is an estimate of the importance of the premise statement in the PTLM's "worldview" relative to that of the other premise statements.

$$P_{NLI}(s_h) := \mathbb{E}_{s_p \sim P_{scaled}}\big[P(s_p \wedge (s_p \implies s_h))\big] = \frac{1}{\sum_{p \neq h} P_{PTLM}(s_p)} \cdot \sum_{p \neq h} P_{PTLM}(s_p) P_e(s_h, s_p))$$
$$P_{NLI}(\neg s_h) := \mathbb{E}_{s_p \sim P_{scaled}}\big[P(s_p \wedge (s_p \implies \neg s_h))\big] = \frac{1}{\sum_{p \neq h} P_{PTLM}(s_p)} \cdot \sum_{p \neq h} P_{PTLM}(s_p) P_c(s_h, s_p)) \tag{3}$$

4.4 Scoring and Correcting the Predictions

Using one of the three described methods, we obtain estimates for $P(s_h)$ and $P(\neg s_h)$ for each s_h. Ideally, $P(s_h) = 1 - P(\neg s_h)$, but this is rarely the case in practice since the NLI has not perfectly learned logical relationships among all statements. Thus, we average the two types of estimates to get a final NLI-based estimate of these probabilities. Lastly, we use a hyperparameter λ to balance these NLI-based estimates with the original PTLM probabilities to obtain a final, adjusted estimate of $P(s_h)$, which act as our "scores" for each statement:

$$\texttt{score}(s_h) := \lambda\big(0.5 \cdot P_{NLI}(s_h) + 0.5 \cdot (1 - P_{NLI}(\neg s_h))\big) + (1 - \lambda) P_{PTLM}(s_h) \tag{4}$$

With this scoring scheme, we then apply an iterative improvement approach to correct the PTLM's original predictions. We invert ("flip") the lowest-scoring statement if it is under a minimum score threshold, recompute all the statement scores taking into account the new revised answer, and repeat until no more scores are under the threshold or a maximum number of flips is reached. Both the min score threshold and the max number of flips permitted are hyperparameters we fine-tune with the development BeliefBank dataset (Section 5.3).

4.5 Using a Constraint Solver

The above-described approach attempts to use NLI probabilities to adjust the predicted probability of each statement within a batch and to correct the predictions accordingly. While this is an interpretable and straightforward approach for correcting a batch of PTLM predictions, the use of a greedy iterative correction process raises the question of whether the correction process can be better optimized. We thus also investigate a different approach: using the NLI and PTLM probabilities to transform our model's task into a MaxSAT problem that can be optimized by a constraints solver algorithm. This method would be similar to that of Kassner et al., but rather than manually supplying curated constraints to the constraint solver [6], we would use the NLI probabilities to define the constraints.

In this context, a MaxSAT problem takes as input a set of boolean variables and logical constraints with associated weights, with the objective being to assign true/false values to each variable such that the sum of the weights of the *satisfied* constraints is maximized. Specifically, we define the following MaxSAT problem with a hyperparameter $\alpha \in [0, 1]$:

- For each statement s_i, define a variable x_i and define the following two constraints:
  1. (x_i) with weight $= (1 - \alpha) \cdot P_{PTLM}(s_i)$
  2. $(\neg x_i)$ with weight $= (1 - \alpha) \cdot (1 - P_{PTLM}(s_i))$
- For each pair of statements (s_h, s_p) where $h \neq p$, define the following constraints with weights set to their corresponding NLI estimates:
  1. $(\neg x_p \vee x_h)$ with weight $= \alpha \cdot P_e(s_h, s_p)$
  2. $(x_p \vee \neg x_h)$ with weight $= \alpha \cdot P_e(s_p, s_h)$
  3. $(\neg x_p \vee \neg x_h)$ with weight $= \alpha \cdot P_c(s_h, s_p)$
  4. $(x_p \vee x_h)$ with weight $= \alpha \cdot P_c(\neg s_h, \neg s_p)$

Note that this requires two passes of statements through the NLI model, one pass with the set of positive facts ($s_1, s_2, ...s_n$) and one pass with the set of negative facts ($\neg s_1, \neg s_2, ...\neg s_n$). For each statement s_i, our model predicts it to be true if and only if the statement's corresponding variable x_i is assigned `true` by the constraint solver algorithm.

The hyperparameter α can be interpreted as analogous to the λ hyperparameter used in our iterative improvement approach, as its role is to also balance the NLI probabilities (the constraints weights) with that of the PTLM probabilities (the variable weights), although it is not directly comparable given that the NLI and PTLM signals are processed differently.

5 Experiments

5.1 Data

We are using the BeliefBank dataset curated by Kassner et al. to tune and evaluate our model [6]. The dataset contains a manually defined subset of the ConceptNet [7] semantic knowledge graph (KG) represented as a directed graph with 1,846 nodes and 4,058 edges. Edges capture directional implications between nodes, and we preprocess the graph such that nodes are modeled as statements of the form `(<relation>,<target>:<truth>)`. For example, the edge `(IsA,crustacean:1)` $\rightarrow$ `(IsA,invertebrate:1)` codifies the constraint "If {x} is a crustacean, then {x} is an invertebrate". The relations of the constraint graph nodes are one of: "IsA", "HasA", "MadeOf", "PartOf", "HasProperty", and "CapableOf", which are 6 relations of interest captured by the BeliefBank paper [6]. During preprocessing, we also augment the graph to capture the contrapositives of existing constraints.

The BeliefBank dataset contains 1,072 "calibration silver facts" harvested from the constraint graph about 7 different entities of interest (e.g., "ant", "cypress"). We use this as a development set to fine-tune our model hyperparameters. The BeliefBank test dataset consists of 12,636 factual statements derived from the constraint graph for 85 selected plant and animal entities, while excluding entities that could be ambiguous like "bat" [6].

5.2 Evaluation method

Our goal is to maximize the number of correct answers while minimizing the number of these answers that logically contradict each other. Kassner et al. report F1-score over accuracy score due to class imbalance in the dataset [6]. Since we design for balance in the labels of the batch of facts created for each entity (Section 4.1), we report both F1-score and accuracy score.

Each batch of facts has a set of constraints our model is attempting to satisfy. Let $C(S)$ denote a set of logical constraints over statements S in the batch, where we consider each individual constraint c_i where the premise s_p is true, denoted $\{c_i|s_p\}$. Consistency is defined as the complement of the fraction of all violated constraints, as shown in Equation 5:

$$\texttt{Consistency} = 1 - \frac{|\{c_i \mid \neg(s_p \rightarrow s_h)\}|}{|\{c_i \mid s_p\}|} \tag{5}$$

The numerator is the number of constraints that are violated, i.e., where $s_p \rightarrow s_h$ is false.

5.3 Experimental details

5.3.1 HuggingFace Models

Our model uses the `macaw-large` PTLM (the same one used by Kassner et al. [6]) and `roberta-large-mnli` for NLI with their default configurations. These models are only used in their evaluation states; our approach does not require training or fine-tuning. Our evaluation baseline is `macaw-large` with no additional post-processing.

5.3.2 Hyperparameter Tuning for Scoring and Iterative Improvement

To select optimal hyperparameter values for our scoring and iterative improvement methods (Section 4.4), we run our model on batches of size $n = 50$ sampled from the development BeliefBank dataset (Section 4.1). We tune three hyperparameters:

1. Minimum Score Threshold: Statements whose score (Equation 4) is below this threshold are candidates for flipping the original PTLM answer
2. Max Number of Flips: The maximum number of flips to be applied for a single batch
3. λ: Interpolates between the raw PTLM score and the additional signal from the NLI

Hyperparameters are tuned using the `hyperopt` framework, which leverages the Tree of Parzen Estimators (TPE) [8] algorithm for performing Bayesian Optimization over the search space. We use negative F1 score as an objective function loss to minimize and perform 100 iterations during tuning. Each iteration evaluates a different subset of hyperparmeter values according to the TPE algorithm. Using the tuned hyperparameter values, we then evaluate our model on batches of size $n = 100$ sampled from the test Beliefbank dataset.

On a GCP n1-standard-8 using a Nvidia Tesla V100 GPU, our evaluation against the development set, including hyperparameter tuning, completes in $\approx$ 1.5 hours. Evaluation against the test set takes $\approx$ 10 minutes.

5.4 Constraint Solver Experiments

We use the νZ extension of the Z3 algorithm [9] to optimize the MaxSAT problem we construct for each batch of predicted statements. The runtime required to execute this algorithm can be significantly large to get an optimal answer, especially when handling $O(n^2)$ number of constraints. Thus we cap each run of the Z3 algorithm at 1 minute when running on a GCP instance with a Tesla T4 GPU, after which the algorithm can yield a partial, candidate solution.

In addition, to reduce the complexity of this MaxSAT representation and to mitigate some of the potential noise that could come from summing over a large quantity of many small weights, the constructed MaxSAT problem would only include constraints that have an NLI-predicted probability of at least ϵ. Hyperparameter tuning for α and ϵ was done with a basic grid search approach on batches of size $n = 100$ sampled from the development BeliefBank dataset, with the selected values being $\alpha = 0.91$ and $\epsilon = 0.2$. We then used these values to evaluate our constraints solver approach on batches of size $n = 100$ sampled from the test Beliefbank dataset.

5.5 Results

	Hyperparameters			Metrics		
Method	**Min. Score**	**Max Flips**	λ	**F1**	**Accuracy**	**Consist.**
Baseline	-	-	-	0.787	0.822	0.826
Max	0.573	9	0.422	0.807	0.844	0.836
Average	0.543	6	0.519	0.812	0.85	0.846
Weighted Avg.	0.367	7	0.832	0.833	0.87	0.858

Table 1: Iterative Improvement Approaches vs. Baseline Performance

While all the P_{NLI} approaches we explored outperform the baseline vanilla PTLM in all metrics, the difference in the optimal hyperparameters is significant. The weighted average approach (Equation 3) achieves the best performance with the most strict parameters; fewer iterative flips are needed and the score threshold is much lower. These results confirm our expectation that increasing logical consistency on a state-of-the-art PTLM generally dovetails with increasing accuracy. We expected the max approach (Equation 1) to outperform the baseline by a higher margin since it accounts for the strongest entailment and contradiction probabilities from the batch, but the tuned low λ relative to the other approaches suggests this NLI signal is not discerning enough (see Section 6.3 for further analysis).

Our preliminary experiments with using the constraints solver to resolve NLI-predicted logical inconsistencies within a batch of facts show some promising results, as our best run ($\alpha = 0.909, \epsilon = 0.2$) obtained a F1 value of 0.829, which is significantly higher than the baseline and outperforms the average and max iterative improvement approaches. This approach also achieved a consistency of 0.896, which is much higher than any of the three iterative improvement approaches.

6 Analysis

Question	PTLM	PTLM Prob.	Orig. Score	Output	Final Score	Label
Is an american bison a mammal?	Yes	0.9998	0.7579	Yes	0.7958	Yes
Is an american bison a bird?	Yes	0.8197	0.249	No	0.5053	No
Is an american bison a country?	Yes	0.8096	0.2416	No	0.4002	No
Does an american bison have a face?	Yes	0.9965	0.5095	Yes	0.5233	Yes
Batch mean			**0.5005**		**0.5225**	
Batch variance			**0.0220**		**0.0183**	

Table 2: Select "american bison" inputs and outputs, $n = 100$; weighted average approach

6.1 Increasing Statement Score

We take a closer look at the model's effectiveness in Table 2 by investigating its handling of the motivating american bison example from Section 1. The PTLM answer to "Is an american bison a bird?" causes the affirmative but false statement "An american bison is a bird" to be input to the NLI model. The low initial statement score calculated according to Equation 4 is indicative of this statement being strongly contradictory to other higher-confidence statements such as "An american bison is a mammal". By flipping the original PTLM answer, our model achieves a higher statement score for the statement "An american bison is not a bird". The statement score for "An american bison is a mammal" also increases because the corrected answer results in a higher entailment probability $P_{NLI}(s_h)$ and lower contradiction probability $P_{NLI}(\neg s_h)$ where $s_h =$ `An american bison is a mammal`. The average score increase for the entity as a whole illustrates that our scoring heuristic in Equation 4 effectively identifies incorrect and inconsistent predictions as it was designed to do. Furthermore, the correcting of the "Is an american bison a country?" prediction suggests that our approach of evaluating natural language inference within a batch helps sift out noise that a word like "american" may introduce in this question.

6.2 Hyperparameters

The hyperparameter λ interpolates between the raw PTLM probabilities and our NLI-based statement scores, and has arguably the most dramatic affect on our results. By fixing `min-score-threshold` and `max-num-flips` hyperparameters to their optimal values based on our tuning against the development set (see Section 5.3.2), we can explore varying levels of λ and how they affect the overall accuracy (F1) and consistency for each of our three different aggregation methods.

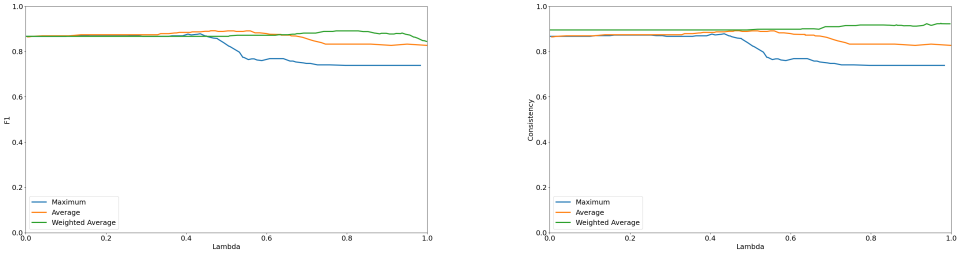

Figure 2: λ influence on: F1-score (Left), consistency (Right)

From Figure 2 we observe that in both F1 and consistency, nearly all approaches result in no predictions being flipped until reaching a $\lambda \approx 0.5$ threshold. At $\lambda = 0.5$, there is considerable divergent behavior between all three aggregation approaches. In particular, as λ starts favoring the NLI-based statement scores, the maximum aggregation approach inadvertently makes incorrect flipping decisions, due to it overestimating both entailment and contradiction probabilities. Empirically, we see that incorporating the probability of premise s_p being true and taking that into account when scoring statements does seem to "soften" the scoring so that the model makes more measured decisions about which statements to "flip".

6.3 Comparison of Methods

Studying confusion matrices of the three P_{NLI} approaches' results offers deeper insight into differences between them. Table 3 contains the confusion matrix for the weighted average approach (Equation 3), and Table 4 corresponds to the maximum approach (Equation 1). Table 5 for the average approach (Equation 2) is in the Appendix.

		Answer Flipped	
		Yes	No
PTLM Incorrect	Yes	446	1067
	No	39	6948

Table 3: Weighted Avg. Confusion Matrix

		Answer Flipped	
		Yes	No
PTLM Incorrect	Yes	390	1123
	No	206	6781

Table 4: Maximum Confusion Matrix

One striking result is that the weighted average approach has substantially fewer false positives - a prediction that was flipped to the incorrect answer - compared to the average approach, and especially the maximum approach. Table 7 in the Appendix comparing the mean and variance of statement scores before and after the iterative correction procedure offers an explanation for this. The variance of the original statement scores computed using a weighted average for P_{NLI} is an order of magnitude higher than the variance observed using the maximum approach. Combining this fact with the comparatively lower minimum score threshold and better metrics implies that the weighted average is better able to separate out predictions that need flipping from already correct predictions. The wider range of statement scores likely allows the hyperparameter tuning to also find a better balance between NLI and PTLM signals since there is more room for nuance between predictions. Despite flipping many fewer statements, the weighted average approach achieves significantly higher precision (0.92) than the maximum approach (0.65). The maximum approach may be overly greedy when determining P_{NLI} and not accounting for the noise inherent in the predictions. Table 7 further reinforces the stronger performance of the weighted average approach by showing that the increase in statement score after iterative flipping is more than double the increase provided by the next-best approach. Given a group of questions that produce similar outputs from the PTLM, the weighted average allows our methodology to rely more heavily on signal from the entailment and contradiction probabilities, whereas the standard arithmetic mean does not. Recall from Section 4.1 that our batches are biased towards containing more premise - conclusion logical implication pairs than would be present if the silver facts were randomly sampled. This essentially makes it easier to make NLI predictions since the statement pairs passed into it likely have a non-neutral relation. Therefore, the weighted average outperforms the arithmetic mean because it places a higher weight (λ) on a strong NLI signal.

6.4 Scoring Heuristic vs. Constraints Solver

We can again examine the confusion matrix (Table 6 in the Appendix) to see differences in how the constraints solver performs when compared to our iterative improvement approaches. Namely, there is a higher number of false positives, which makes sense since the constraints solver algorithm is more likely to find these false positives as it attempts to exhaustively explore all possible true/false combinations of the statements. In contrast, our iterative improvement approaches are more conservative, greedily flipping only one statement at a time and flipping only when the statement score is under a threshold for a limited number of times. Fine-tuning these hyperparameters does help, but for any given experiment run, the selected hyperparameter values are used for every batch, even though the optimal hyperparameter values to use for each batch may significantly differ from batch to batch. From our results, it appears that the more conservative heuristic approach performs better in accuracy while, interestingly, the less conservative constraints solver performs better in consistency, although further experimentation is needed to verify this robustly.

7 Conclusion

In this project, we demonstrated the usefulness of an NLI model for correcting batches of predictions made by a PTLM. In particular, although an NLI can only approximate the probabilities of a given entailment or contradiction between two statements, its output can still be used along with the PTLM's prediction probabilities to identify logical inconsistencies and correct predicted statements

accordingly. In addition, an NLI model's entailment and contradiction probability estimates can be represented as soft constraints in a MaxSAT problem that, once optimized by a constraints solver, can also yield improved predictions that are more accurate and logically consistent. The primary advantage of our work is achieving an accuracy and consistency increase over the baseline without needing manually curated logical constraints specified. However, there are some limitations to our approach warranting further research. We biased our batches towards facts that are more likely to have a strong entailment or contradiction relation with each other, which resulted in stronger NLI signals. The benefit afforded by NLI may be less pronounced in a random selection. In addition, there is more room to explore methods for robustly addressing how the NLI model itself may provide logically inconsistent predictions (e.g. when $P_{NLI}(s_h) \neq (1 - P_{NLI}(s_h))$. Furthermore, we evaluate NLI within a batch of facts, which is limited by batch size. Kassner et al. utilized a global memory to store facts [6], which also has scalability limitations, but there may be an optimal balance to be struck. Lastly, further work can investigate mechanisms to better train for consistency in the pretraining of language models, which would reduce the need for additional architecture around them that seeks to correct their mistakes.

References

[1] Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? *CoRR*, abs/1909.01066, 2019.

[2] Nora Kassner and Hinrich Schütze. Negated LAMA: birds cannot fly. *CoRR*, abs/1911.03343, 2019.

[3] Oyvind Tafjord and Peter Clark. General-purpose question-answering with macaw. *CoRR*, abs/2109.02593, 2021.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[5] Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. A logic-driven framework for consistency of neural models, 2019.

[6] Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. Beliefbank: Adding memory to a pre-trained language model for a systematic notion of belief, 2021.

[7] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, 2016.

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[9] Nikolaj Bjørner and Anh-Dung Phan. νz-maximal satisfaction with z3. *Scss*, 30:1–9, 2014.

A Appendix

A.1 Confusion Matrices

		Answer Flipped	
		Yes	No
PTLM Incorrect	Yes	361	1152
	No	123	6864

Table 5: Avg. Confusion Matrix

		Answer Flipped	
		Yes	No
PTLM Incorrect	Yes	721	792
	No	304	6683

Table 6: Constraint Solver Confusion Matrix

A.2 Summary Statistics

Method	**Orig. Score Mean**	**Orig. Score Variance**	**Final Score Mean**	**Final Score Variance**
Max	0.735	0.0083	0.7386	0.0073
Average	0.6918	0.0114	0.6991	0.0097
Weighted Avg.	0.5415	0.0198	0.5594	0.0169

Table 7: Summary Statistics Comparison for Scoring Methods