IntrospectQA: Building Self-reflecting, Consistent Question Answering Models

Stanford CS224N Custom Project

Maya Srikanth
Department of Computer Science
Stanford University
`msrikant@stanford.edu`

Rachael Wang
Department of Computer Science
Stanford University
`rachaelw@stanford.edu`

Abstract

Large pretrained language models (PTLMs) embed vast amounts of data and language patterns in their parameters, achieving state of the art performance on a variety of NLP tasks. Yet, these models often suffer from a fundamental weakness: they struggle to maintain logical consistency. We aim to improve the logical consistency of large pretrained language models (PTLMs) in the question answering (QA) setting as follows: (1) augment a pretrained QA model with an external memory for storing past model predictions and (2) integrate supervisory signals from a large pretrained NLI model to encourage consistency between past and future QA model predictions. Existing approaches for enforcing logical consistency rely on constraint solving algorithms like MAXSAT, which operate on hand-engineered constraint graphs specifying implications between facts: by utilizing a powerful NLI model to enforce consistency, our approach, IntrospectQA, removes the burden of constructing extensive constraint graphs and provides for generalization beyond a finite set of predefined logical clauses and entities. Our preliminary experiments indicate that IntrospectQA can boost performance over a baseline QA model over a wide variety of topics without any pretraining, finetuning, or external constraint graphs, suggesting that leveraging a pretrained NLI model is a potential avenue for improving the logical consistency and accuracy of a QA model.

1 Key Information to include

- Mentor: Eric Mitchell

2 Introduction

Pretrained language models (PTLMs) embed vast amounts of data and language patterns in their parameters, achieving state of the art performance in a variety of NLP tasks, from machine translation and language modelling, to question answering. Despite well-publicized successes with PTLMs, recent works demonstrate that these models have a fundamental weakness: they suffer from inconsistent beliefs, in a logical and semantic sense, and contradict prior correct answers after sufficient probing [1][2]. For instance, a PTLM optimized for a True-False question answering task may correctly answer “True" to the questions "Is a peahen a female bird?" and "Can female birds lay eggs?", but incorrectly answer “False" to the question “Can a peahen lay eggs?". Such fallacies indicate that PTLMs struggle to maintain a coherent "mental model" of the world, and lack a deeper understanding of language patterns and factual entities embedded in their parameters [1]. Even after targeted finetuning to reduce inconsistencies, PTLMs still logically collapse after sufficient questioning [2].

In this project, we aim to improving the logical consistency of large pretrained models (PTLMs). We take the following definition logical consistency: if a model assigns “True" to the clauses X and

Y, and X and Y both imply clause Z, then the model should assign “True" to clause Z.[1]. To the best of our knowledge, one prior work specifically addresses the issue of logical inconsistency in pretrained QA models [1] by augmenting a QA model with an external model memory “BeliefBank" and a constraint-solver algorithm. While this approach improves QA model consistency and accuracy over time, it (1)relies on the existence of a hand-engineered constraint graph specifying logical relationships between various clauses, and (2) is confined to the finite set of topics included in the constraint graph. We explore whether we can utilize a pretrained NLI model to boost logical consistency and F1 accuracy for a pretrained QA model, while removing dependency on external data and providing for generalization beyond a finite set of topics.

On a high level, our approach, IntrospectQA, is as follows:

1. Augment a large, T5 pretrained Macaw QA model [3] with an external memory for storing past model predictions
2. Integrate supervisory signals from a large pretrained Roberta NLI model [4] to encourage consistency between past and future QA model predictions

We present a novel algorithm for utilizing NLI predictions to refine QA model knowledge over time (Algorithm 1). Furthermore, we present a batching algorithm to generate ample opportunity for self-contradiction over an evaluation run, thereby providing a challenging evaluation setting for IntrospectQA (Algorithm 2). We also evaluate and report IntrospectQA performance on a challenging subset of BeliefBank data, a collection of facts on various entities and the logical implications between these facts [1].

3 Related Work

The published work "BeliefBank: Adding Memory to a Pre-Trained Language Model for a Systematic Notion of Belief" is an effort to build more logically consistent, rational NLP agents without model retraining or finetuning. Authors add a novel memory layer called a "BeliefBank" on top of Macaw, a PTLM (pretrained T5 Macaw QA model) to track model beliefs over time and modify raw PTLM answers to improve consistency[1]. The authors offer two mechanisms to maintain a systematic notion of belief throughout a session of "interaction" with the PTLM: (1) a reasoning component (a weighted MaxSAT solver) which revises model beliefs that conflict with prior beliefs or predefined logical constraints, (2) a feedback component which sends new queries as input to the PTLM, including prior model beliefs as context for these queries. Note that the MaxSAT solver is an efficient approximation algorithm for the NP-complete weighted maximium satisfiability task, defined as follows: given some CNF formula with weights assigned to each clause, find the truth assignment to the CNF formula variables which maximizes the sum of weights of the satisfied clauses [5]. Closely related to this work is research on knowledge graph identification, a task commonly addressed by using a constraint solver to produce a maximally consistent knowledge graph given noisy entities and relations [6]. Our approach differs from this work since we use a pretrained NLI model (instead of a MAXSAT solver) to encourage consistency while removing the burden of constructing constraint graphs and providing for generalization to a wider variety of topics.

Closely related to the task of improving logical consistency is prior literature exploring knowledge consistency for question answering on cloze (“fill in the blank") tasks that have different paraphrasings, but are semantically equivalent [7]. In particular, the authors propose a novel consistency loss and demonstrate that it better equips BERT to achieve higher consistency on unseen relations. Our work is different since we (1) focus on logical consistency rather than knowledge consistency, (2) employ post-processing techniques with an NLI model to improve QA model performance rather than altering the QA model training objective.

While the technique of utilizing an external memory that we and other authors [1] use may seem reminiscent of retrieval-augmented generation techniques, it differs from architectures like RAG and REALM since model memory is built from prior outputs instead of external sources [1].

To place this work in a broader context, our project and previous approaches for improving QA model consistency are part of a larger effort in NLP to build more robust, intelligent PTLMs with the capacity to maintain accurate mental models. While the experimental setting of our project and previous

[1]Credit to Eric Mitchell for defining this

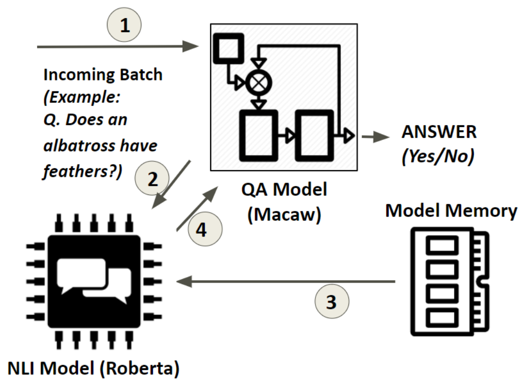

Figure 1: IntrospectQA Framework.

relevant work is limited to a True-False (or yes-no) QA task on a finite set of entities, constraints, and relations, we suspect that the framework of enforcing logical consistency using persistent global memory could be extended to more extensive NLP tasks, from more complex question answering to dialogue generation.

4 Approach

4.1 Definitions

First, we define terms that are vital for understanding BeliefBank data [1] and our algorithms; *entity, constraint, condition, conclusion,* and *consistency.*

In the context of this project, *entity* refers to any of the 85 animals or plants present in the BeliefBank dataset, described in more detail in Section 5.1. Note that BeliefBank data [1] is comprised of constraints and facts. A *constraint* is a logical relationship between two *facts* and their associated truth values, .T (true) or .F (False). For our setting, constraints come in 2 forms [1]: **(1)** positive implications (e.g. “X is a dog.T $\rightarrow$“X has a tail.T") and **(2)** mutual exclusivities (e.g. “X is a dog.T $\rightarrow$ X is a bird.F "), where X is a placeholder for any entity of interest. Importantly, a *constraint* is comprised of a *condition* and *conclusion*, where the *condition* implies the *conclusion*. For instance, in the example for positive implication, “X is a dog.T" is the condition, while “X has a tail.T" is the associated conclusion.

Finally, let's motivate definition of the *consistency* performance metric. First, note that a constraint c_i can be expressed as a 5-tuple of the form $c_i = (s_i.l_i \rightarrow s_j.l_j, w_i)$, where s_i, s_j are sentences extracted from BeliefBank entities and facts, $l_i, l_j \in \mathit{True}(T), \mathit{False}(F)$, and w_i denotes the weight of the constraint c_i. More concretely, this constraint indicates that if s_i has truth value l_i (denoted $s_i.l_i$), it follows that s_j has truth value l_j, that is, $s_j.l_j$ [1]. Then, for all constraints $c_i \in C(S)$, we have the following definition of *inconsistency* τ [8]

$$\tau = \frac{|\{c_i|\neg(s_i.l_i \rightarrow s_j.l_j)\}|}{|c_i|s_i.l_i|} \tag{1}$$

That is, τ is the size of the set of violated constraints divided by the size of applicable constraints. This brings us to the authors' definition [1], *consistency* $= 1 - \tau$. In words, *consistency* is 1 minus the fraction of constraints whose *condition* is *believed* (e.g. the condition is correctly predicted by the QA model), but whose *conclusion is not* (e.g. the conclusion is not correctly predicted). Note that a low consistency indicates that the QA model tends to logically contradict itself.

4.2 IntrospectQA Architecture

Figure 1 shows our IntrospectQA framework, which operates as follows. First (see **(1)**), a batch of yes-no questions is issued as input to a Macaw QA model. Next, the Macaw QA model generates predictions for this batch, and these predictions (and their associated questions) are added to an external model memory. After this, (see **(2)**, **(3)**) a Roberta NLI model [4] is used to generate "contradiction", "entailment", "neutral" log probabilities for all pairs of premises and hypotheses (e.g. all pairs of declarative sentences formed from questions and associated QA model predictions) in external model memory. Algorithm 1 describes how we leverage these NLI model probabilities to revise (or preserve) a QA model prediction for a given hypothesis in model memory. Specifically, for any hypothesis in model memory, [2] we compute the sum of contradiction log probabilities between this hypothesis and all other premises [3] in model memory such that each log probability we sum is ≤ -0.0015 (denoted sum(max_contr)). We repeat the same procedure for entailment [4] log probabilities produced by the NLI model between the given hypothesis and all other premises in model memory (denoted sum(max_entail)). For the revision step, the general intuition is as follows: if there are more premises in model memory which have a high probability of contradicting the given hypothesis than there are premises which have a high probability of entailment with the given hypothesis, then we have cause to invert the original QA model prediction for the given hypothesis (e.g. yes $\rightarrow$ no, or no $\rightarrow$ yes).

4.3 Algorithms and Methods

Algorithm 1 NLI Strategy 1

```
Compute latest batch of predictions with $\hat{y}$ =**QA_model**$(x)$ = $argmax_y p(y|x)$
Add latest batch $(x, \hat{y})$ to model_memory
revised_predictions ← [ ]
for each hypothesis, hyp_pred in model_memory: do
    max_contr, max_entail ← [ ], [ ]
    for each premise, prem_pred in model_memory: do
        hyp_NLI ← format (hypothesis, hyp_pred) into a declarative sentence
        prem_NLI ← format (premise, prem_pred) into a declarative sentence
        contr_logprob, neutral_logprob, entail_logprob ← append NLI_model(prem_NLI, hyp_NLI)
    end for
    max_contr ← contr_logprob ≥ -0.0015
    max_entail ← entail_logprob ≥ -0.0015
    if $sum$(max_entail) < $sum$(max_contr) then:
        $y_{new}$ ← ¬ hyp_pred
    else
        $y_{new}$ ← hyp_pred
        revised_predictions.append($y_{new}$)
    end if
end for
return revised_predictions
```

Our novel ConflictBatcher class (Algorithm 2) is designed to challenge IntrospectQA (Algorithm 1) by setting traps for logical inconsistency in a streaming evaluation setting, where IntrospectQA processes several batches of questions sequentially and optionally refines prior predictions after processing each batch. On a global level, for any evaluation round, we'd like to provide IntrospectQA ample opportunity to contradict itself across batches. To that end, as Algorithm 1 indicates, ConflictBatcher assumes that every evaluation round focuses on a specific entity e: by progressively probing IntrospectQA on a specific entity, the probability of self-contradiction over time is much higher than it would be if we were to include logically independent questions focused on different entities. On a more granular level, we'd like to provide IntrospectQA an opportunity to contradict itself while processing a given batch of input. To that end, every batch is constructed by first sampling a condition from the list of conditions associated with a given entity e, and then sampling 2 conclusions associated with this condition, repeating until the batch is built. Since IntrospectQA sees 2 conclusions for any

[2]In this context, a "hypothesis" represents a BeliefBank yes/no question and associated QA model prediction, which is formatted into a declarative sentence rather than a question and answer

[3]Like hypothesis, premise is another Beliefbank yes/no question and prediction, formatted into a declarative sentence

[4]A high entailment probability indicates that the given hypothesis follows logically from the given premise

condition within a batch, it could potentially contradict itself by correctly predicting a condition, and incorrectly predicting either of the associated conclusions.

Algorithm 2 CONFLICT BATCHING (each run focuses on a entity e)

```
conds←list of conditions corresponding to entity $e$
concl←list of conclusions corresponding to entity $e$
1. Sample condition $c_i$ from conds
2. Sample 2 conclusions $a_i$ from concl such that $c_i \rightarrow a_i$ is specified by a constraint
Repeat (1), (2) until batch is built
```

5 Experiments

5.1 Data

BeliefBank is a dataset with 4998 simple facts (concerning 85 entities (plants, animals)) and $12,147$ constraints specifying logical implications between various relation types (see the list below) and associated truth values [1]. We use the BeliefBank dataset constraints and "silver facts" evaluation subset to construct a baseline evaluation set of roughly 5486 questions concerning 85 entities [1]. To construct the data, we map constraints and "silver facts" from the BeliefBank dataset [1] to questions. Our preprocessing implementation maps various relation types (along with relevant entities) to yes-no QA model inputs, as demonstrated below. Note that the QA task for any factual question about an entity is to correctly predict "yes" or "no".

1. **IsA:** albatross, (IsA, bird: yes)$\rightarrow$ **Q.** Is an albatross a bird? **A.** Yes.
2. **HasA:** albatross, (HasA,feathers: yes) $\rightarrow$ **Q.** Does an albatross have feathers? **A.** Yes.
3. **HasPart:** albatross, (HasPart,face: yes) $\rightarrow$ **Q.** Does an albatross have a face? **A.** Yes
4. **CapableOf:** albatross, (CapableOf,fight for life: yes) $\rightarrow$ **Q.** Can an albatross fight for life? **A.** Yes.
5. **HasProperty:** albatross, (HasProperty,alive: yes) $\rightarrow$ **Q.** Is an albatross alive? **A.** Yes.

Importantly, for this experiment, we use only questions that *align with some constraint*: that is, the question is either **(1)** the condition of some constraint whose answer is the constraint-specified truth value for some entity or **(2)** the conclusion of some constraint, whose answer is the constraint-specified truth value. To make this more concrete, consider the hypothetical constraint "X is a female bird".T $\rightarrow$ "X lays eggs".T, as well as the entities "hen" and "dog". Using this constraint, an example of question pair *we would include* is (1)"Is a hen a female bird?" and (2) "Can a hen lay eggs?": this is because the "hen" entity *does satisfy* the first condition in this constraint, i.e., it is a female bird. However, *we would not include* the question pair (1) "Is a dog a female bird?" and (2)"Can a dog lay eggs?": this is because the entity "dog" *does not satisfy* the condition of this constraint, as it is not a female bird. By selecting only questions such that the entity and associated fact align with some constraint, we avoid introducing unnecessary noise and potentially inflating performance with blunt questions with obvious yes-no answers.

5.2 Evaluation method

We measure 2 metrics, (1) accuracy (in terms of F1 score with respect to the true answers) and (2) consistency, defined as $1 - \tau$, where τ is the fraction of constraints whose condition is correctly predicted but whose conclusion is incorrectly predicted. A low consistency score indicates that the model tends to contradict its past predictions. To replicate the BeliefBank paper evaluation procedure, we report F1 and consistency as a function of the number of batches the QA model has processed. Importantly, after processing each batch, both accuracy and consistency are recalculated over the *entire* model prediction memory (including the latest batch).

As Algorithm 2 indicates, each evaluation "run" issues questions focused on a specific entity (e.g. BeliefBank animal or plant) to both IntrospectQA and a Macaw QA baseline. As we recompute both consistency and F1 accuracy over the entire model memory after each new batch is processed, each "run" results in a consistency curve and an F1 accuracy curve. For each metric, we average curves over all entities in a given evaluation round, and report these averaged curves.

5.3 Experimental details

We evaluate whether IntrospectQA can outperform a Macaw QA baseline on BeliefBank yes/no constraint-aligned questions across a variety of configurations (# entities, # batches, and batch sizes): figures 2, 3, 4, and 5 show the resulting average metric curves and specify average performance improvements. Note that we use both Macaw QA and Roberta NLI off-the-shelf, without any pretraining or finetuning. More specifically, we use Macaw-Large, a 770 million parameter pretrained T5 transformer model, as our QA baseline: this is the same QA model used in the closely related BeliefBank paper [1].

5.4 Results

Quantitative Analysis. Preliminary experiments, described in Figures 2, 3, 4, and 5, suggest that IntrospectQA can use NLI entailment and contradiction signals to aptly revise the baseline Macaw QA model yes-no predictions on constraint-guided questions from BeliefBank [1] data, outperforming the baseline Macaw QA model on average in terms of F1 accuracy and consistency. We present results over a variety of configurations which vary the number of batches, batch size, and the number of entities evaluated.

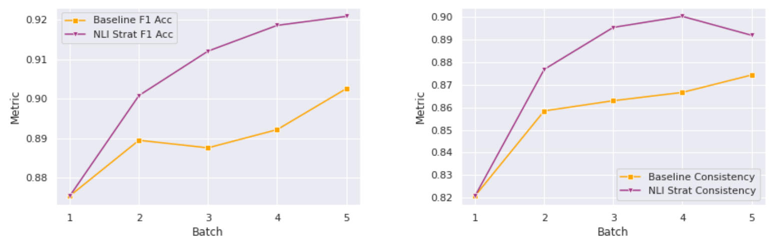

Figure 2: Average metric curves over 6 entities (with at least 100 questions each) over 5 batches of size 15. IntrospectQA offers performance boost of $+1.8\%$ average F1, and $+2.4\%$ average consistency.

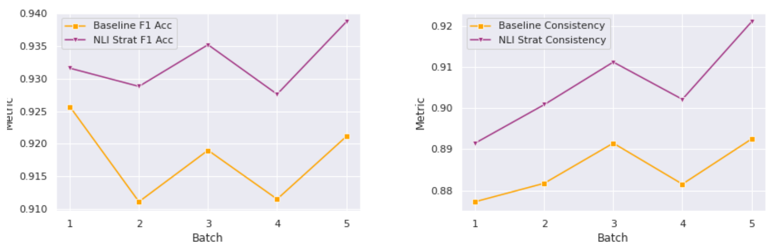

Figure 3: Averaged metrics curves over 11 entities (with at least 70 questions each) over 5 batches of size 13. IntrospectQA offers performance boost of $+1.6\%$ average F1, and $+2.3\%$ average consistency.

6 Analysis: IntrospectQA Strengths & Limitations

6.1 Strengths: Improved Average Performance

As shown in figures 2, 3, 4, and 5), preliminary results over a variety of experimental configurations indicate that augmenting a QA model with an NLI model (and using the strategy described in Algorithm 1) has the potential to improve QA model performance in terms of F1 score and accuracy by using new context to revise prior knowledge. In particular, we observe these performance improvements when we *average* metric curves across evaluation rounds featuring various distinct entities, with batch sizes ranging from $10 - 15$. We evaluate over subsets of entities of sizes ranging from 6 to 74 of the 85 entities provided in BeliefBank [1].

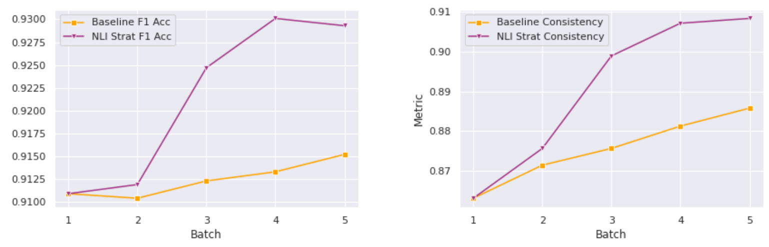

Figure 4: Averaged metrics curves over 35 entities (with at least 60 questions each) over 5 batches of size 10. IntrospectQA offers performance boost of $+1\%$ average F1, and $+1.7\%$ average consistency.

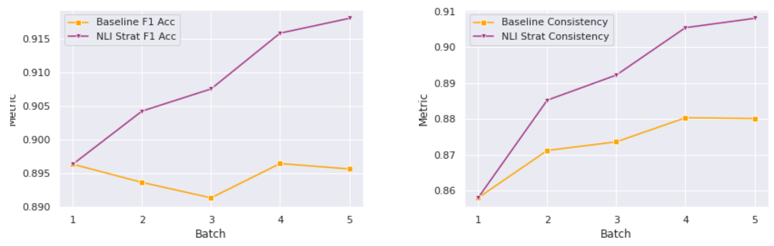

Figure 5: Averaged metrics curves over 74 entities (with at least 50 questions each) over 5 batches of size 10. IntrospectQA offers performance boost of $+1.5\%$ average F1, and $+2.0\%$ average consistency.

6.2 Limitations: Threshold Hyperparameter and Entity-level Performance

IntrospectQA performance is fairly sensitive to the setting of the threshold parameter, $t = -0.0015$. For instance, when t is increased to -0.002, the model is more likely to unnecessarily revise predictions for both NLI strategies (see Algorithm 1). However, preliminary experiments suggest that the threshold of $t = -0.0015$ is associated with improved performance across a broad range of topics and experiment configurations: it's possible that minimal tuning is required to optimize the current threshold for other topics of interest.

While IntrospectQA consistency and accuracy metrics are higher than baseline QA model metrics when averaged over all entities in a given evaluation run (see Figures 2, 3, 4, 5), we find that this trend does not necessarily hold at the entity level (see Figure 6), which suggests that our approach is highly sensitive to sampled data in a given evaluation run. That is, if IntrospectQA is exposed to a small number of noisy examples, it's possible that the model can incorrectly alter prior knowledge for a specific entity. This failure case indicates that while NLI model can indeed improve average performance across a variety of topics, it's likely that more sophisticated objectives are required to enforce performance improvements on a more granular, topic-level scale. It's unclear whether a MAXSAT-based approach is susceptible to similar issues, as the authors of the Beliefbank paper [1] do not provide entity-level results.

6.3 Noisy Comparisons with BeliefBank Algorithms

While authors of the BeliefBank paper [1] also aim to improve Macaw QA logical consistency on BeliefBank questions (albeit with a MAXSAT solver-based approach), their performance figures are not directly comparable to ours, for several reasons. Firstly, our baseline Macaw model significantly outperforms their reported Macaw baseline scores: our model achieves an average $F1 \approx 0.90$ and consistency ≈ 0.87 in a streaming setting with over 5000 questions, compared to their $F1 \approx 0.69$ and consistency ≈ 0.72 in a streaming setting. This trend extends to the entire dataset: our Macaw baseline QA achieves $F1 \approx 0.77$ and consistency ≈ 0.892 on the entire set of silver facts. These discrepancies are likely due to differences in preprocessing, evaluation data, and batch sampling. For instance, as the authors do not specify their preprocessing pipeline, it's plausible that our method for

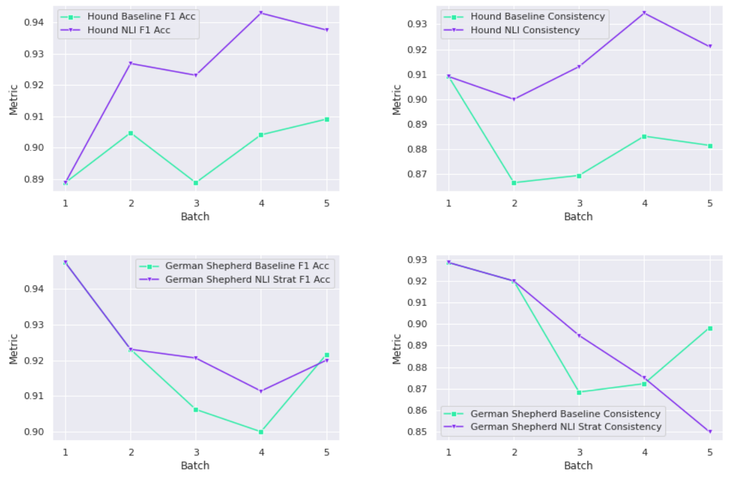

Figure 6: **Top row:** Success case at the entity level, for "hound": IntrospectQA outperforms baseline Macaw QA model in terms of both accuracy and consistency. **Bottom row:** Failure case at the entity level, for "german shepherd": IntrospectQA achieves lower final consistency than baseline, due to over correction.

mapping relations to questions is more sophisticated and grammatical than theirs, thereby improving QA model performance. Further, while the authors evaluate their models on the full subset of $12,500$ silver facts, we evaluate over questions which align with some constraint. Finally, while the authors do not specify their batch sampling procedure in their evaluation, we proactively sample batches such that there is (i) significant opportunity for self-contradiction within a given batch sampled during a "run" focused on one entity, and (ii) significant opportunity for self-contradiction across batches in a given run, as a single run is focused on "probing knowledge" on one entity. It's possible that this adversarial batch construction makes for a more daunting challenge than the authors' streaming setting: in any given batch, IntrospectQA is guaranteed to encounter "traps" for logical inconsistency. Finally, due to compute constraints and limits on the number of constraint-guided questions relevant to each entity, our batch sizes range from ≈ 10 to 15, much smaller than their 1200: this may help explain the higher variance of our metrics.

While we cannot precisely compare the two methods, perhaps it is still informative to roughly compare numbers, albeit with the reservation that this comparison is significantly noisy. For instance, both our baseline and NLI strategy achieve final F1 scores in the low to mid .90's, outstripping the best-performing BeliefBank-augmented model, which achieves a max $F1$ score of 0.86. Further, in a number of runs, IntrospectQA achieves final consistency scores in the low to mid .90's, outperforming 2 **non** MAXSAT-based strategies: these figures are not too far behind their MAXSAT-based approaches (consistency ≈ 0.96), even though we use no external constraint graphs.

7 Conclusion

This project was an opportunity to evaluate whether an external memory, paired with supervisory signals from the NLI model, will allow us to improve the consistency and accuracy of a large, pretrained QA model. We implement novel preprocessing and batch sampling, as well as a novel NLI model strategy to evaluate this framework in the setting of question-answering for True/False (yes/no) questions. Preliminary experiments suggest that strategically leveraging NLI model outputs can improve *average* logical consistency and F1 accuracy in a streaming setting on any topic, but more sophisticated objectives are required to manifest this performance improvement on a more granular, topic-level scale. Future steps for this project include refining IntrospectQA's strategy, potentially removing heuristic thresholds in favor of learnable parameters, and adding objectives to optimize for entity-level performance.

References

[1] 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.

[2] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models, 2021.

[3] Oyvind Tafjord and Peter Clark. General-purpose question-answering with macaw, 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. *CoRR*, abs/1907.11692, 2019.

[5] Roberto Battiti. *Maximum satisfiability problemMaximum Satisfiability Problem*, pages 2035–2041. Springer US, Boston, MA, 2009.

[6] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, pages 542–557, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[7] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and Improving Consistency in Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 12 2021.

[8] Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. A logic-driven framework for consistency of neural models. *CoRR*, abs/1909.00126, 2019.