LIQMA: Natural Language Inference to correct Question-answer Model Assertions

Stanford CS224N Custom Project

Patrick Liu
Department of Computer Science
Stanford University
`pliu1@stanford.edu`

Kevin Yang
Department of Computer Science
Stanford University
`kgyang@stanford.edu`

Ian Ng
Department of Computer Science
Stanford University
`iyhn8192@stanford.edu`

Abstract

In this project, we aim to improve language model consistency in binary question-answering tasks by identifying and negating contradictory assertions via natural language inference and a variety of correction algorithms. To do so, we apply a pretrained Question Answering (QA) model to judge whether or not statements about a chosen entity are true, and generate assertions based on those judgments. We then apply a Natural Language Inference (NLI) model to the produced statements in pairwise fashion, determining the probability of QA assertions being contradictory, entailed, or consistent with one another. Due to QA models' incomplete knowledge base or their inability to interpret queries, they generate a significant proportion of incorrect assertions, which often contradict with correct assertions. Our goal is therefore to use the outputs and confidences of the QA model, as well as the contradiction probabilities output by the NLI model, to determine which assertions should be negated to minimize contradictions and thus improve consistency. We implement several different families of algorithms to determine which judgments to negate; these methods fall under the categories of either constraint satisfiability problems, contradiction- or entailment-based probabilistic estimation, and requerying of the QA model.
Full code is available at `https://github.com/patrickliu2011/cs224n-project`

1 Key Information to include

- TA mentor: Eric Mitchell
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Pretrained language models, or PTLMs, are trained on large datasets of real-world information. With this large amount of data, PTLMs store an extreme amount of knowledge; however, language models often have difficulty acting according to consistent beliefs – that is, when queried about logically linked statements, they will sometimes return contradictory answers. For instance, a model might respond to the question “Is a sunflower a plant?” with “Yes,” as well as to the question “Do plants perform photosynthesis?” with “Yes,” but might respond to “Do sunflowers perform photosynthesis?” with “No,” an inconsistent answer given its responses to the first two questions.

As a result of this lack of consistency, it is often difficult to determine what a model actually “believes” or to enforce consistent, reasonable behavior in a model. This is especially important in applications of PTLMs as knowledge bases: a knowledge base would be pointless if inconsistent! However, PTLMs otherwise have promising potential as knowledge bases: they promise substantial potential improvements upon traditional knowledge bases, including the ability to query and understand the organization/structure of knowledge through natural language.

One of the major hurdles to overcome for a PTLM to serve as a knowledge base is thus improvement in consistency. Consistency is difficult to achieve, however, partially because PTLMs tend to memorize entities and relations present in training data. Therefore, they lack an ability to extend symbolic reasoning (e.g. after having seen that kangaroos are mammals and that mammals have hair, they should infer that kangaroos have hair) or to understand unseen words/phrases [1]. PTLMs also often exploit spurious statistical patterns and cues in training data, rather than capturing the semantic meaning of text, the latter of which is necessary to “understand” and capture it as knowledge [2]. Attempts to use language models as knowledge bases must address these issues in sometimes unconventional ways to improve the consistency of PTLMs.

With this in mind, our goal is to improve the usability of a PTLM as a knowledge base by analyzing its beliefs through generating assertions which demonstrate those beliefs. By comparing assertions to each other, we can determine inconsistencies, which signify incorrect beliefs on the part of the PTLM, and can furthermore apply algorithms to determine assertions that are likely to be incorrect. Doing so allows us to correct those assertions during post-processing of the PTLM’s beliefs, improving the its accuracy and thus its consistency over its own beliefs. To do so, we will attempt to detect inconsistent pairs of beliefs/assertions, determined from PTLM queries, using an NLI model which has been pre-trained to this specific task. By using various constraint satisfaction, probability-based, and requerying approaches to correct assertions, we achieve a significant improvement both in the PTLM’s consistency over said beliefs and the accuracy of those beliefs.

3 Related Work

Using PTLMs to serve as knowledge bases is an exciting prospect, and one that has been explored extensively in recent years. Indeed, studies on large PTLMs, such as BERT-large, have found success in utilizing PTLMs as knowledge bases, demonstrating that PTLMs, even without fine-tuning, already encode accurate relational knowledge comparable to other standard NLP approaches with oracle knowledge [3]. Inconsistencies within PTLMs, furthermore, can give us valuable insights into issues in PTLMs’ knowledge structure and representations of concepts. However, despite the importance of making PTLMs consistent, the training process for PTLMs does not typically involve any mechanism to remove inconsistencies, as noted in [4]. Indeed, as Elazar et al. 2021 demonstrate, even large PTLMs such as BERT- and RoBERTa achieve at maximum a consistency of 70% when tested on queries that are identical except for differences in syntactical structure. The same paper demonstrates that it is possible to introduce a specialized consistency loss to add onto the pre-training process; however, while this can increase consistency, it can sometimes come at the cost of accuracy– an equally important metric when it comes to establishing PTLMs as successful knowledge bases.

One approach to improving consistency is to insert a PTLM into a more broad structure that also includes an evolving, symbolic memory of beliefs, a “BeliefBank,” that maintains the consistency of its outputs with its beliefs [5]. The original BeliefBank paper achieves this via a MaxSAT (constraint) solver that negates binary beliefs that contradict significantly with others, as well as a feedback mechanism that poses new questions using existing beliefs as context. By doing so, this system requires no finetuning of the PTLM. It still achieves both accuracy and consistency gains with an external mechanism, thus improving controllability and interpretability as compared to a black-box language model. The BeliefBank paper also contributes a dataset containing a set of 2612 weighted constraints (e.g. “X is a mammal” $\rightarrow$ “X has hair”) and manually annotated facts (e.g. “An american bison is a mammal”), which we make use of in our study. However, one key limitation of the BeliefBank system is that it necessitates hand-written constraints to evaluate consistency, meaning that it would be difficult to generalize this model even in a relatively simple "yes-no" binary setting. To remedy this, as discussed, we employ adversarially trained NLI models to determine entailment, contradiction, and neutrality scores between pairs of statements; this allows us to efficiently determine, with high accuracy and without the effort of having human-written constraints, the most likely relations between statements.

4 Approach

The PTLMs which we test in this paper produce **assertions** in response to yes-no queries. For instance, when queried "Is a buffalo an animal?", a PTLM will either respond with the correct assertion "A buffalo *is* an animal", or the incorrect assertion "A buffalo *is not* an animal". These assertions represent the PTLM's beliefs about said queries. Our main goal is to test various **correction algorithms** to determine, for a set of binary assertions about a given entity, the assertions which should be negated to maximize consistency with the other assertions and the overall accuracy. Our overarching pipeline works as follows:

1. **Data Sampling:** For 10,000 batches, select an entity (e.g. "computer") with a list of ten associated **facts** per batch (e.g. "A computer is a machine."). More details on the sampling process are provided in the "Experiments" section.
2. **QA Querying:** Convert each fact to a binary (yes-no) question (e.g. "Is an computer a machine?") and query a pre-trained question-answer (QA) model [6] on each question. Using the results from the QA model, create a set of 10 binary (yes-no) assertions (e.g. "Yes" $\Rightarrow$ "A computer is a machine."; or "No" $\Rightarrow$ "A computer is not a machine"), as well as the **confidence** the QA model has in those assertions. *Note that the QA model in question is the PTLM whose performance we will be evaluating.*
3. **NLI Contradiction Scoring:** Use a RoBERTa-based NLI model [7], pretrained on determining consistencies between every pair of assertions, to calculate a 10x10 **matrix of contradiction/entailment scores** representing how likely two assertions are to be entailing or contradictory.
4. **Correction Algorithms:** Using the values in the NLI matrix and the QA confidence values, use various algorithms (details provided in section 5) to determine which assertions are false, and **negate** the incorrect assertions to correct them.

In addition to testing the various correction algorithms, we perform a **performance survey** over three different QA models: the Macaw-large QA model [6], the Aristo RoBERTa-QA model [1], and the UnifiedQA (T5)-small model [8]. The base contradiction rates are lowest for the RoBERTa model, and highest for UnifiedQA. The RoBERTa model also has the highest accuracy, while UnifiedQA has the lowest accuracy We compare the metrics for these three models against each other (discussed in section 5.)

Baseline: As a simplistic baseline, we can simply determine accuracy as measured as the average number of correct assertions generated over all 10,000 batches, without utilizing any correction algorithms or doing any negations. We establish baseline values for all three models over all metrics; full baseline scores can be viewed in the Results section.

5 Experiments

5.1 Data

To generate facts associated with entities, we sample from the BeliefBank[5] dataset, which contains a set of 85 entities and 2,612 hand-written constraints of the form "statement 1 implies statement 2". These direct constraints can be used to create a graph of implications between statements about entities, where statements are represented as nodes and the relations between their truth values are represented as edges. For instance, the statement "X is a dog" might be a node with a directed connection to the node "X is a bird", connected by the constraint T, F, since if the statement "X is a dog" is true, the statement "X is a bird" is false. We can therefore represent this constraint as a statement: "A dog is not a bird." Using these statements, we can traverse the graph

[1]`https://huggingface.co/LIAMF-USP/aristo-roberta`

to generate new constraints, using logical implications of the form $A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow C$. For instance, if we have the statement "A border collie is a dog" and "A dog is not a bird", we can generate the new constraint "A border collie is not a bird." Using this method, we augment our dataset to 15,277 directional constraints by chaining together consecutive edges in the graph.

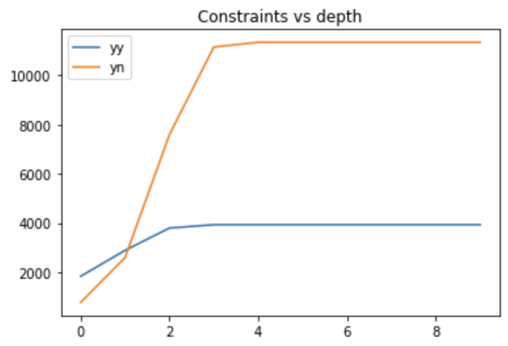

Figure 1: A graph of the number of constraints over time, by depth. For instance, a derived constraint $A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow C$ has depth 2, and $A \rightarrow B \wedge B \rightarrow C \wedge C \rightarrow D \Rightarrow A \rightarrow D$ has depth 3. A `yy` constraint has form $A \rightarrow B$, and a `yn` constraint has form $A \rightarrow \neg B$.

From this augmented set of constraints, we first select at random batches of $N = 10$ entities (e.g. "buffalo," or "dog"). We then sample 10 facts per entity. Using the facts as input, we then follow the pipeline process described in the previous section.

We use a development set containing 65 randomly selected entities of the 85 available, and their relevant constraints, to develop our correction algorithms and and to tune correction algorithm hyperparameters. Our experiments are performed on the held-out test set, the remaining 20 unseen entities and their constraints.

5.2 Evaluation method

We evaluate our correction algorithms on the following metrics, measured *after all negations have been executed*:

Accuracy is the proportion of of assertions that correlate to true facts in our dataset, out of the total number of assertions.

Consistent + Accurate measures the proportion of assertion pairs within a batch that are both correct and non-contradictory, out of the total number of assertion pairs in the batch.

Contradictions the ratio of contradictory assertion pairs to relevant assertion pairs. Relevant answer pairs are the subset of pairs which constraints determine as either directly contradictory or entailing.

Negations is the proportion of assertions negated by the correction algorithm.

F1 score is the geometric mean of recall and precision. Recall in this case is the ratio of the number of actual contradictions negated by an algorithm to the the number of actual contradictions originally present; precision is the ratio of the number of actual contradictions negated by an algorithm to the number of contradictions (actual as well as incorrectly modified) negated by the algorithm.

5.3 Experimental details

As discussed earlier, we test several different classes of correction algorithms, and perform hyperparameter tuning on each.

1. **Constraint satisfiability problems:** In this approach, we construct a boolean MaxSAT (maximum satisfiability) problem by creating a weighted graph of constraints, based on the NLI and QA model outputs. We have two forms of soft constraints:

- Unary constraints, provided as a single boolean assertion (e.g. "A dog is an animal" is `True`) with the constraint's weight being the assertion's confidence from the QA model's output.
- Binary constraints, with the constraint's weight generated from the confidence that the inverse of pairs of assertions are contradictory, given by the NLI model's output. This is necessary to convert the contradiction likelihood between two assertions into conjunctive normal form. For instance, if assertions A: "A dog is an animal" and B: "A dog is not a mammal" have a high contradiction score of 0.9, a CNF constraint of $\neg A \vee \neg B$ is added with a small weight of $1 - 0.9 = 0.1$, because $\neg A \vee \neg B \Leftrightarrow \neg(A \wedge B)$.

We then utilize the RC2 MaxSAT solver [9] to find the assignment of truth values to our assertions that maximizes our weights; the assertions for which truth values are `False` are the assertions which get negated. This approach was inspired by the BeliefBank's use of a MaxSAT constraint solver. Our hyperparameter tuning experiment for the relative weighting between unary and binary constraints found the optimal (binary) contradiction score scale factor to be $3\times$ the (unary) constraint weight.

2. **Contradiction-based probabilistic estimation:** In this family of approaches, we examine the NLI and QA outputs corresponding to each individual assertion to determine the probability of a given assertion being a contradiction based on its agreement with other assertion. We have ten different expressions for doing so. In the below expressions, take S to be the set of assertions about a given entity, $c(a)$ to be our confidence that assertion a should be negated; $N(a, b)$ the NLI model's confidence that a and b contradict, and $Q(a)$ the QA model's confidence in assertion a.

C1. We calculate $c(a) = \frac{1}{\|S\|-1}\sum_{b \in S \setminus a} N(a, b)Q(b)$, Then, if $c(a) > \lambda_1 \cdot Q(a)$, we negate a. In essence, this is a weighted average of the contradiction values of a with other assertions. If a contradicts with a assertion b that we are more confident is true, the chance that a needs correction is higher; conversely, if we do not have high confidence in the truth of b, the fact that a contradicts with b is not as important.
We note the optimal hyperparameter via finetuning for the negation threshold $\lambda_1 = 0.4$.

C2. We calculate

$$c(a) = \frac{1}{\|S\|-1}\sum_{b \in S \setminus a}\Big(\frac{Q(\neg a)Q(b)}{Q(\neg a)Q(b) + q(a)Q(\neg b)}N(a, b) + \frac{Q(\neg a)Q(\neg b)}{Q(a)Q(b) + q(\neg a)Q(\neg b)}(1 - N(a, b))\Big)$$

and negate a if $c(a) > \lambda_2$. Here, we are essentially calculating the weighted probability of a being false given that a and b contradict, or that a and b agree, and totaling them. We note the optimal hyperparameter via finetuning for the negation threshold $\lambda_2 = 0.4$.

C3. We use the same procedure as in expression C1, but use the threshold $c(a) > \lambda_3$ instead of $c(a) > \lambda_1 \cdot \mathrm{Q}(a)$. We note the optimal hyperparameter via finetuning for the negation threshold $\lambda_3 = 0.5$.

C4. We calculate $c(a) = \frac{1}{\|S\|-1}\sum_{b \in S \setminus a}(N(a, b) - 0.5)Q(b)$ and negate a if $c(a) > 0$; this centers the contradiction probability around 0 instead of 0.5.

C5. We calculate $c(a) = \frac{1}{\|S\|-1}\sum_{b \in S \setminus a}(N(a, b) - 0.5)(Q(b) - 0.5)$ and negate a if $c(a) > 0$, centering both the contradiction probability and confidence at 0 instead of 0.5.

C6. We calculate $c(a) = N(a, \max_{b \in S \setminus a}(Q(b)))$, which determines the contradiction probability of a in comparison to the single assertion $b \neq a$ with the highest confidence. Then, if $c(a) > \lambda_6$, we negate a. We note the optimal hyperparameter via finetuning for the negation threshold $\lambda_6 = 0.9$.

C7. We use the same procedure as in C6, but negate a if $c(a) > 0.7$ (arbitrarily selected).

C8. We construct set $C = \{(a, b) | N(a, b) > 0.5\}$. Then, for each $c \in C$, if $Q(a) < Q(b)$, we negate a; otherwise, we negate b, essentially selecting the lower confidence assertion of each contradictory pair and treating it as requiring correction.

3. **Entailment-based probabilistic estimation:** In this family of approaches, we utilize the NLI's *entailment score*: the NLI model's confidence that assertion a entails assertion b (note

that this is a directional relation.) Letting $N_e(a, b)$ denote the entailment score for a to b, we develop the following algorithms:

C9. We first utilize one of the contradiction probabilistic estimation approaches (in practice, C1, as it appears to perform the best) to find an initial set of potential corrections, denoted as B. Then, for each $b \in B$, we find all assertions a such that $N_e(a, b)$ is high (with a threshold of $\lambda_9 = 0.7$, obtained via hyperparameter finetuning). We negate a, because if b requires negation, it is likely that a also requires negation.

C10. For every pair of assertions (a, b), we query the NLI model to find pairs for which there is a high entailment score $N_e(a, \neg b)$ (with a threshold of $\lambda_{10} = 0.7$, obtained via hyperparameter finetuning). Then, for every such pair, if the QA model's confidence in b is low, we negate b.

4. **Requerying approach:** In this approach, we first utilize a contradiction-based probabilistic estimation (once again, using C1 in practice). Once we have determined the set of potential correction targets, we then add an additional layer of filtering on top. This filtering comes in the form of "requerying" the QA model; that is, reformulating the assertion as various forms of a question and querying the QA model once again on the reformulated question. The intuition here relies on the fact that QA models' response and confidence is not necessarily robust to wording changes in a query, even if the core aim of the query remains the same. Therefore, by requerying, we can effectively increase the average quality of our QA model's outputs by selecting the more confident of the two outputs, under the assumption that the more confident output tends to correspond to the more correct assertion.

C11. In this approach, as stated above, we determine the set of assertions which C1 determines to require correction. For each such assertion a in the set, we then iterate through every other assertion $b \neq a$ in the original batch, generating a new question q with b as context and a as the query. For instance, if we were examining the assertion "A buffalo is a bird", we might iterate to the assertion pair "A buffalo is a bird." and "A buffalo has four legs."; our generated query will then be "A buffalo has four legs. Is a buffalo a bird?" We then take the QA model's average confidence over all such generated queries corresponding to $a - b$ pairs that the NLI model determines as contradictory. If that average confidence exceeds our initial confidence in assertion a, we negate a.

C12. Once again, we determine the set of assertions which $C1$ determines to require correction. For each such assertion in the set, we reformulate it, as well as its negation as a question. For instance, for the assertion, "A buffalo a bird.", we would generate "Is a buffalo a bird", as well as "Is a buffalo not a bird?" We then requery the QA model on each question pair, and select the response with higher confidence as our final verdict on the original assertion. This reasoning follows from [10], which finds that the relations between negation pairs are often misunderstood by PTLMs. It therefore stands to reason that if a QA model has more confidence in the negation of an assertion than in the original assertion, the negation is understood better and is therefore more reliable.

5.4 Results

We report, over the span of all correction algorithms and all three QA models in the model survey, the results on all metrics, in Figure 2.

We can observe that C12 (the negation-based requery correction algorithm) most consistently performs the best across all metrics; however, this must be taken into account with the fact that the requery correction algorithms are built upon algorithm C1, which already narrows down the set of potential corrections. On the other hand, correction algorithm C8 performs extremely poorly according to all statistics; it also happens to be by far the algorithm that makes the most negations. The other probabilistic estimation approaches perform reasonably similarly, though C1 tends to outperform the rest (hence, we use it as the basis for the requery and entailment-based algorithms.) In terms of our model performance survey, we can note that overall, the Aristo-RoBERTa model generally performs superior on all metrics to the Macaw-large model, which in turn performs superior to the UnifiedQA model.

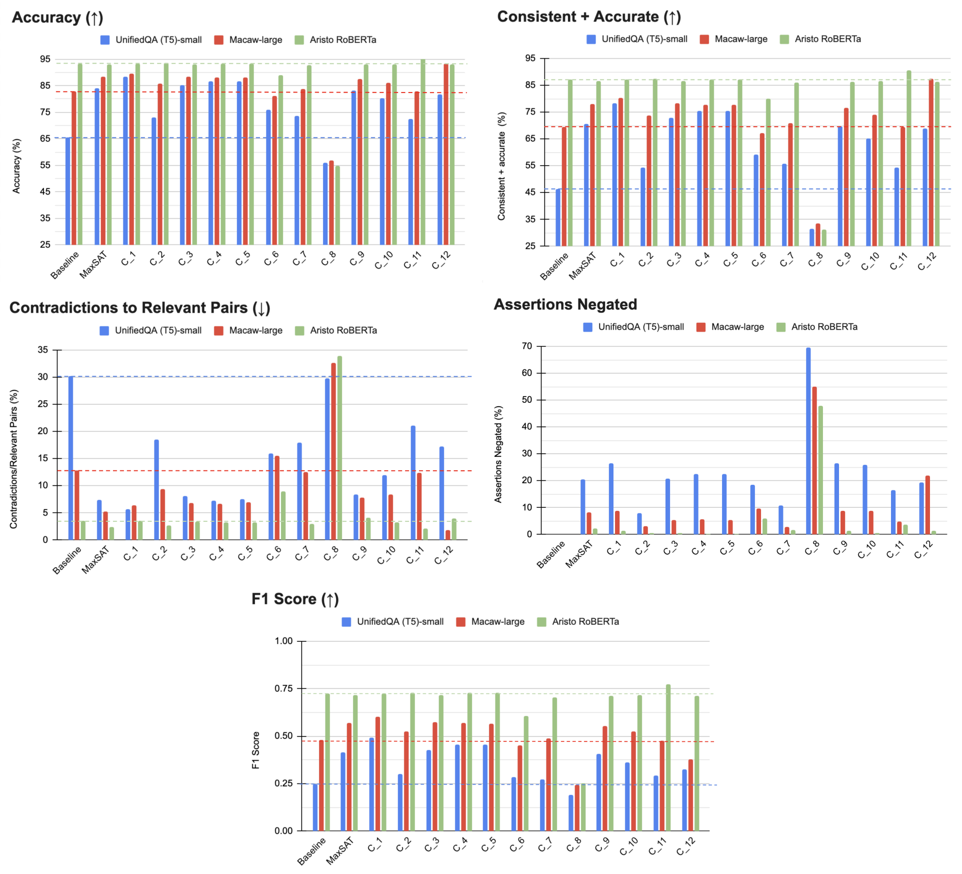

Figure 2: Graphs of the metrics, compared for all corrections and all QA models. The horizontal dotted lines represent baseline measures for all three models in each metric. In the titles of each graph, ↑ represents that higher values are more optimal; ↓ represents that lower values are optimal. "Assertions Negated" isn't necessarily optimal in any direction, though in practice values towards the extremes tend to be suboptimal.

6 Analysis

We can analyze with respect to each of the different classes of algorithms (MaxSAT, C1-C8, C9-C10, C11-C12). We note the following observations:

The MaxSAT solver results in consistently lower contradiction rates compared to the other methods, other than C12. On the other hand, its accuracy falls behind some of the other methods such as C1. Since the MaxSAT solver performs near-optimal constraint satisfaction with respect to our hyperparameters, the confidence and contradiction scores may not directly correspond to the optimal unary and binary weights for the algorithm, so our MaxSAT may not achieve optimal performance with respect to our final objective. Furthermore, because the NLI model outputs do not necessarily match with the BeliefBank constraints, the optimal approach given the input is not necessarily optimal relative to the evaluation objective.

Contradiction-based probabilistic estimation methods tend to perform approximately the same, and consistently better than the baseline. The exception is C8, which seems to negate assertions far more than necessary. Out of these methods, C1-C5 perform the best. This may be attributable to the fact that they incorporate all pairs involving a given assertion, as well as utilizing both all QA confidence weights and all NLI contradiction scores. On the other hand, C6-C8 only compares assertions with the most confident assertion in each batch as the reference, which is highly dependent on the accuracy of the highest-confidence assertion and misses contradictions that do not involve the highest-confidence assertion. As a theme which should be intuitive, these probabilistic estimation methods perform better the more comparisons and confidences they are provided with. To this end, it's possible that with more statements per batch, we could achieve a higher overall accuracy; however, this may come at the cost of significantly increased runtime.

The entailment-based methods C9-C10 perform similarly to some of the contradiction-based methods, C4-C5. This is likely because the method in which we used entailment scores to negate statements seemed intuitively but may not be entirely theoretically sound. Given more time, we would have liked to investigate entailment-based methods further.

The requerying approach C11, which requeries using other assertions as context, performs poorly compared to the other models in most of our experiments. The exception is in our QA model performance survey with the Aristo RoBERTa V7 model, where C11 outperforms the other models significantly.

As stated in the results, C12 is the model that performs consistently the best. We hypothesize that this is because language models are sensitive to question phrasing such as negation, even though the question content is the same. As a result, questions that yield lower-confidence incorrect assertions may be corrected by higher-confidence correct assertions.

We consider our results across the RoBERTa based model, Macaw-large, and UnifiedQA. The correction algorithm performances generally follow the same trends for each model; it seems that despite the differences in QA model accuracy and consistency, the algorithms have a fairly clear ordering of performance. One main exception to this rule is with the RoBERTa-based model, where C11 works better and C12 performs slightly worse relative to their performance with the other QA models. We believe this may have to do with different QA models having different errors and confidences when handling question context and negation, which specifically impact the behavior of C11 and C12 compared to the other correction algorithms.

7 Conclusion

Overall, we find that in practice contradiction-based probabilistic correction algorithms can perform similarly to theoretically optimal approaches like MaxSAT solvers for yes-no question correction. Moreover, requerying approaches that use different question wordings and contexts are a promising approach to correction model performance. We also find that the relative performances of these correction algorithms are dependent on the density of contradictions in the QA model predictions.

In the future, we would like to develop more methods like C12, which address the issue of model inconsistency across different paraphrasing. For example, instead of using a single question, we may pass in multiple versions of the question from a paraphrase generator, and use the outputs to get a more phrasing-independent prediction. We would also like to experiment with learned models, such as reinforcement learning algorithms with the prediction vector as a state and individual negations or requeries as actions. Lastly, we may want to try more explicitly graph-based algorithms using the NLI-based factor graph

In terms of testing our existing correction algorithms, we would like to perform experiments across different NLI models and see how different pretraining corpora impact performance on our dataset. Although surveying across our QA models varies the density of contradictions in each batch, we would also like to try testing batches with extremely high contradiction density, in order to see if multi-step correction algorithms like MaxSAT begin to outperform single-step algorithms like C1.

We note that our experimental setup uses very limited question topics and phrasings. Thus, to make this study more applicable to realistic situations, we would want test our algorithm performance with more unconstrained questions. This includes binary questions that are not phrased as yes-no questions (and, in the longer term, questions that aren't necessarily binary), statements with different sentence structures, and queries about different entities than merely provided in the BeliefBank data.

References

[1] Nora Kassner, Benno Krojer, and Hinrich Schütze. Are pretrained language models symbolic reasoners over knowledge?, 2020.

[2] Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational*

Linguistics, pages 4658–4664. Association for Computational Linguistics, July 2019.

[3] 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.

[4] 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, 2021.

[5] Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. Beliefbank: Adding memory to a pre-trained language model for a systematic notion of belief. *CoRR*, abs/2109.14723, 2021.

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

[7] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[8] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single QA system. *CoRR*, abs/2005.00700, 2020.

[9] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. Rc2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11:53–64, 09 2019.

[10] Arian Hosseini, Siva Reddy, Dzmitry Bahdanau, R. Devon Hjelm, Alessandro Sordoni, and Aaron C. Courville. Understanding by understanding not: Modeling negation in language models. *CoRR*, abs/2105.03519, 2021.