Introduction

In this project, we aim to improve language model consistency in history question-answering tasks by identifying and flipping contradictory statements via natural language inference and a variety of correction algorithms. To do so, we apply a question-driven natural language inference (QDNL) model to judge whether or not statements about a given entity are true. We then apply a natural language inference (NLI) model to the produced statements in pioneer fashion, determining the probability of QA judgments being contradictory, entailed, or consistent with one another.

Due to the NLI model’s incomplete knowledge base or its inability to interpret the given statements, the QA model generates contradictory judgments in 31% of the time. Our goal is therefore to use the outputs and confidence of the QA model, as well as the contradictions probabilities output by the NLI model, to determine which judgments should be regarded to minimize contradictions and thus improve consistency. We implement several different families of algorithms to determine which judgments to flip.

Background: Data, Sampling, and Models

• Entities, facts, and constraints are drawn from the BioBertBank memory bank. The original BioBertBank provides 60 entities, 5,252 facts, and 2,253 constraints of the form “statement A implies statement B.” We augment these constraints according to the following logical expression: A ⇒ B ⇒ C ⇒ A ⇒ C.

• Repeatedly applying this chain logic, we can augment over the course of about 10 iterations to 15,277 constraints.

• The bio set contains 66 entities and corresponding facts/constraints drawn from BioBertBank. The test set contains the other 25 entities.

• Statements are sentences of the form “entity [text] [contextual description].”

• The QA model takes in statements, converted to the form of a binary (true/false) question. It outputs its answer to the question, as well as its confidence in the provided answer.

• The NLI model ranks in a list of statements of the above form. For each pair of statements (a, b), it provides a triple (a, b, c) of probabilities, where a is the probability that c is implied, b is the probability that c is entailment, and c is the probability that a and b are entailments.

Experiments

To detect contradictions, we apply four classes of correction algorithms:

1. Constraint entailment problem: In this approach, we construct a baseline QA system (Baseline entailment problem) by creating a weighted graph of constraints, based on the NLI and QA model outputs. We have two tiers of weights for each constraint: those for the NLI model’s output and those for the QA model’s output. Binary constraints, that is, with constraints’ weights generated from the confidence statement pair are contradictions based on the NLI model.

We then utilize the R2C MuDAT to find the optimal configuration that maximizes constraint weights.

2. Constraint-based probabilistic estimation (C3-CB). In this family of approaches, we examine the NLI and QA outputs corresponding to each individual statement to determine the probability of a given statement being a contradiction based on its agreement with other statements.

Example:

\[ C(a) = \frac{1}{2} \sum_{i=1}^{n} N_i \times Q_i(a,b) \]

3. Entailment-based probabilistic estimation (C3-CB). In this family of approaches, we utilize the NLI’s entailment probability – the probability predicted by the NLI model, that statement a entails statement b (note that this is a directional relation).

4. Forward approach (C3-CF). In this approach, we first utilize a constraint-based probabilistic estimation. Once we have determined the targets for flipping using such a method, we then add an additional layer of filtering, or “refining” the QA model, that is, reformulating the statement to various forms of a question and querying the QA model once again on the reformulated question. Intuition: The QA model’s response and confidence is not necessarily robust to wording changes in a query given similar semantic meaning.

In addition to our correction methods, we perform several ablations, across:

• QA model: Our base QA model, UnilmAR T5-small (performs worse than Macaw), and RoBERTa V7 Model (performs better than Macaw).

• Correction algorithm hyperparameters: For algorithms with hard thresholds for entailment, contradiction or flipping, we varied across several threshold values.

Pipeline and Methods

For our experiments, we follow the following steps:

1. 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.”)

2. Convert each fact to a yes/no question and query a pre-trained Macaw QA model on each question. Convert Yes/No responses to statements.

3. Use a RoBERTa-based NLI model, pretrained on determining inconsistencies between statements, to calculate a 0.5M weights of contradictory scores representing how likely any pair of assertions is to be entailing, or contradictory.

4. Using the values in the matrix and the QA confidence values, identify and correct incorrect statements (described in detail in Experiments).

Conclusions

• Especially favorably performing QA models, using a correction algorithm with confidence-weighed contradiction score can far outperform the QA model by itself.

• By rephrasing the QA model’s output for both positive and negative statements, accuracy can be boosted even higher than with probabilistic estimation methods.

• While our methods yield improvements over baselines for various QA models, the relative performance of our methods is around the same.

• Future approaches to further improve consistency:

• Evaluate graph-based approach

• Compare with other state-of-the-art correction methods.

• Experiments with high confidence single fact batches, and with paraphrasing.

References


CS 2014 Graduation Final Project