Creation of a Dataset for Premise Generation

Eldad Peretz (323820225) and Yuval Nirhod (323976910)

Abstract

This work is about premise generation, which is used for detecting unanswerable questions. Our target in this work is, given a question, to generate its premise. Then one can use a premise correctness classifier to know if the question is answerable. We will attempt to create a meaningful dataset for this problem, one that will allow us to use deep learning based model training in order to generate premises in a more semantic way. We hope that this will allow us to get better results than the previous methods, which are rule based.

1 Introduction

Question answering is the task of, given a question, predicting the answer. Well known QA datasets (for example SQUAD [3]) contains unanswerable questions. An analysis in [1] reveals that 21% of the unanswerable questions can be explained by the false premise problem. The premise of a question is defined by all the facts that one can deduce from the question itself. For example, the premise of "Where are my apples?" is that my apples are somewhere. A more sophisticated question could include two or more premises, for example, in the question "Where are the apples I hid in my beach house?" we have the premise that I hid my apples, but also that I have a beach house.

The problem of false premise is simply that the premise is incorrect. For example, consider the following question, "When did Elon Mask found Microsoft". The premise is "Elon Mask did found Microsoft". This premise is wrong as a well known fact is that Bill Gates founded Microsoft. Another example is "When did Aristo invent the lightbulb?". The premise is again wrong as it's known that Thomas Edison invented the lightbulb. If we query this type of questions in Google search engine we get an answer, although the question is unanswerable and every answer would be wrong. For example, google's search engine answers the question "When did Aristo discover America?" by "October 12, 1492" although Aristo didn't discover America, Christopher Columbus did.

The first step towards identifying false-premise questions, is the task of premise generation. Given a question, we want to output its premise. This task is very difficult, and its main problem is that we do not have a dataset for this task. We assume that the reason for this may be that the task of extracting the premise takes a lot of time and requires linguistic knowledge.
The main goal of our work is to create a meaningful dataset for this task, one that will enable models that train on it to learn premise generation.
We tried to create such dataset in a lot of ways, but in this paper we will present two main methods. We start by presenting the first approach and its disadvantages, and then move to the successful approach which we explain in more detail. We trained a model on both approaches' datasets, to check if the dataset is meaningful, and explain the results.

In the first approach we design by hand many templates of questions and premises, and using WikiData insert information to those templates to create the dataset. In the second we take a QA dataset (SQUAD [3]) and from each question extract the premise in a automatic way which is not neccesarly completely accurate. By comparing with the same benchmark as in [1], our premise generator is state of the art in premise generation and the first work, as far as we can tell, that uses deep learning to generate the premise.

2 Dataset Generation

We present two ways for dataset generation, the first is based on templates and the second is based on parsing. We first define the dataset we want to achieve in a more formal way.

2.1 Dataset definition

In a question there are many premises. For example, in the question "Did the redhead king of England eat an apple?" we have the premise that England has a king, which is also a redhead, and that an apple is some concept in the world which can or cannot be eaten. A lot of these premises are not interesting, and can be extracted easily from the question by recursively going over groups of word which are grammatically connected by way of description or elaboration, and adding their existence in the world as a premise.
We decided to focus on a more complicated type of premises, that derives from more complex relations between the words. We found out that in simple yes or no questions, the premises are not interesting, like in the example we just gave. Therefore we decided to focus on Wh-questions, which yield more complex premises that can derive from the question word and its relations with the rest of the question. We call this type of premise the main premise of the question. The main premise represents the meaning of the question the best.
We decided to focus on the main premise because it is the most difficult one to derive, and after finding it we can find the rest of the premises more easily. Therefore, the dataset we are building will consist of pairs of a Wh question and a main premise.

2.2 Wh questions

A Wh question is a request for information, in which the answer is something more elaborate than a simple yes or no. In this approach, we decided to focus on Wh-questions. The reason is that in Wh-questions the premise of the question derives from the Wh question word and is based a lot on it. For example, in the question "Where did you eat an apple?", the premise is not only the existence of an apple, but that you did eat it somewhere, as opposed to the question "Did you eat an apple?", which only yields the premise that an apple is a concept.

3 Premise Generation

3.1 Conditional Generation

An example of a conditional generation task is document summarization. Given a document of length L, we want to generate a summarization of that document of size $l \ll L$ that preserves the content of the original document. We can also formulate the premise generation task as conditional generation problem - given a question, we produce the premise of the question. The BART model is a transformer, sequence to sequence, large language model [2] that can be used for conditional generation. Previous work [2] showed that BART is capable of a summarization task which is closely related to our task. We show that BART is also capable to perform premise generation. Notice that this result is a contribution to what we know can be extracted from large language models such as BART.

3.2 Finetuning BART

In this work, instead of training a massive model by ourselves, we use BART as a pretrained model and we finetune it. As described, BART is a strong model for conditional generation. In the training we freezed BART pretrained values and optimized only the added layers, so it will generalize better.

4 Templates Approach

Our first thought was that the easiest way to create a dataset of pairs of questions and premises will be by preparing all kinds of templates of questions and their corresponding premises, such that we can insert information to those templates and thus create a multitude of different questions and premises. We decided to use WikiData to extract the information needed to fill the templates.
For example, we can define the pair of templates "When did [Person] invent [Tool]?" (Question) and "[Person] did invent [Tool]" (Premise). We now can collect pairs of (Person, Tool) from WikiData and fill these templates automatically, getting a lot of pairs for our dataset. We can make the templates more complicated, adding more places where information needs to be inserted, and thus making more complex questions and premises.
WikiData provides a very convenient API that can be used to exact this kind of information easily, thus making the task of filling the templates very easy and automatic.

4.1 Disadvantages of this method

The main problem with this approach is that creating the templates is a very difficult task. We couldn't build a template which was flexible enough to generate questions and premises with different structures. Thus for each kind of question we had to build a different template by hand, making the task **very** time consuming. In order to cover half of the possible structures of a question in the English language we estimated that we will have to design hundreds of templates, therefore making this method ineffective.

4.2 Training the Model

Despite the above disadvantages, we designed over 20 templates and fine-tuned BART on the dataset created. The results were not good. When testing the model on questions derived from the templates the model was trained on, it reached OK results, but when tested on different kinds of questions, even with only slight differences which weren't compatible with the templates, the results were close to non-sense, hence proving indeed that we need a lot more templates to enable the model to generalize.

4.3 Conclusions from the experiment

The failure of this approach led us to believe that the key to solving this problem is not generating pairs of questions and premises which we know them to be correct. We searched for another way of creating such dataset, one that may not be 100 percent accurate but will capture all the different kinds of questions to allow the model to generalize.

5 Parser Approach

In this approach, we attempted to semi-solve the problem in an automatic way, and hoped that the model could generalize from there.

We noticed that a lot of the time, the premise could be deducted from the question's type and the relationships between the words in the question. For example, In any "When did something happen?" questions, like "When did Aristo discover America?", we can say that the premise is the "Something" - "Aristo discovered America". Another example is questions like "Who did something?", like "Who discovered America?", where the premise is that someone did this thing - "Someone discovered America". These observations led us to believe that we can solve the premise extraction problem, or at least get pretty good results, using an automatic parser, that relies on Universal Dependencies.

Extracting the Universal Dependencies is a complex task, that requires NLP model too, and thus we hoped that when training a model on a dataset which consists of the parser's output, the model will not learn to behave like the parser, but rather learn how to extract the premise in a way that will generalize better, and hence will achieve better results.

5.1 Universal Dependencies

Universal Dependencies, frequently abbreviated as UD, is an international cooperative project to create treebanks of the world's languages. Dependency is the notion that words are connected to each other by directed links. The verb is taken to be the structural center of clause structure. All other words are either directly or indirectly connected to the verb in terms of the directed links, which are called dependencies.
We believe that for a lot of questions, the premise could be extracted from the relations between the words. UD trees provide a good representation of such relations and hence it made sense to use them.

5.2 The Parser

First of all, given a question, the parser uses Stanford NLP group's Universal Dependencies parser to create the universal dependence tree. Then, it parses the UD tree and that is how we get the premise.
Given the UD tree, the first thing the parser does is to classify what kind of question this is. We divided the questions into 6 different types, and we treat each type differently.
We will now attempt to explain how we divided the questions and extracted the premises. In order to avoid getting into technical details, we will not get into how exactly we extract the premise. instead we will give a general explanations and add examples.

- **Type 1:** This is the case where there is a **Wh-adverb which is connected directly to the root of the question** in the UD tree. For example, in the question "When did Egypt close the Suez Canal to Israel?", we have the Wh-adverb "When" connected to the root "close" in the UD tree. We discovered that in such

cases, we can usually remove the Wh-word, move the subject of the sentence to the beginning, followed by the verb and then the rest of the question, and get the premise. For example, in this case the parser will yield: "Egypt did close the Suez Canal to Israel".
The reasoning behind this decision is that the Wh-word serves as a modifier of the whole question, because it is connected to the root, and thus the premise is the event described.

- **Type 2:** This is the case where there is a **Wh-adverb which is connected to the root of the question, but not directly**. For example, in the sentence "How much did Victoria donate to Ireland to help with famine relief?", the root is "donate", whereas the Wh-adverb "How" is not connected to it, but to the word "much" which modifies the root. To get the premise in this situation, we discovered that it is usually sufficent to remove some of the words that connect the Wh-word to the root, and then continue like type 1. In this case it will yiled "Victoria did donate to help with famine relief to Ireland".
The reasoning here is that in this type of questions, the Wh-adverb is not a stand alone modifier, but a part of a group of words that together modify the root, and the rest follows like type 1. We found out that this group of words is almost always the words that are on the trail between the Wh-adverb and the root in the UD tree.

- **Type 3:** This is the case where there is a **Wh-pronoun which is connected directly to the root of the question** in the UD tree. For example, in the sentence "From whom did ministers derive their power?", the root is "derive", and the Wh-pronoun is "whom". In this case, we switch the Wh-pronoun with some word, according to what Wh word it is, and then shift the order like in type 1. For example, the word "Whom" we switch with the word "Someone", and in this case get "Ministers did derive their power from someone".
The reasoning here is that given that the Wh-word is a pronoun, it will function as the subject of the premise.

- **Type 4:** This is the case where there is a **Wh-determiner connected to the root of the question** in the UD tree. For example, in the sentence "Which coast is Miami located on?", the word "which" is a Wh-determiner and is connected to the root "located". In this case, we switch the Wh-determiner by some word according to what Wh word it is, and then shift the order like in type 1. For example, the word "which" we switch with "some", and thus get "Miami is located on some coast".
The reasoning that led us to this solution is that this kind of Wh-words are modifiers that determine the kind of a reference a noun has. Therefore they need to be replaced with such modifier in the premise.

- **Type 5:** This is the case where **the root of the question is a Wh-pronoun**. For example, in the sentence "What is the capital city of the US?", the word "What" is both the root of the question and a Wh-pronoun. In this case, we discovered we can just switch the root with some word according to the what Wh-word it is, and just leave the sentence as is. For example, we will switch "What" with "Something", and get "Something is the capital city of the US".
The reasoning here is exactly like type 3. (The reason for the two different types is that it effects other parts of the parser, which I will not mention here.)

- **Type 6:** This is the case where **the root of the question is a Wh-adverb**. For example, in the question "When was the battle of Saratoga fought?", the root of the question is the word "When", which is a Wh-verb. In this case, we discovered we can switch the order of the words after removing irrelevant ones, and get the premise. For example, in this case we will get "the battle of Saratoga was fought".
The reasoning here is exactly like type 1. (Again, the separation is needed because of other parts of the parsing.)

5.3 Parser Result Analysis

We tested the parser on 100 Wh questions sampled randomly from our questions dataset. The results are displayed in Table 1. We can see that the parser achieves 62.5% accuracy, and gets up to 73.75% accuracy when counting the Grammar mistakes as correct answers, which is reasonable because those are very small errors that still give a pretty accurate premise.

The reason we split the analysis into those 4 catagories, is because we believe that the model will still be able to learn from the G and S categories, and will fix the mistakes the model makes. Thus we believe that although the parser was not accurate, S and G categories are still good for us.
Overall we think that the parser achieves really good results, especially when considering that the current benchmark gets 80% accuracy.

5.4 Improving the parser

As we can see from Table 1, the parser is far from perfect, and a deeper linguistic analysis of the questions will result in a more accurate premise. For example, in type 2 and type 6 questions, you can see that our parser has a really hard time outputing the exact premise, and almost always makes some small mistake. (Because the mistake is more often than not minor, we can still use it in training, but we assume the results will be better with a better parser)

6 Results

6.1 Model training on parser dataset

We fine-tuned BART on our parser dataset, which consists of around 56,000 pairs of questions and premises, which were generated from the parser. We noticed that if we trained the model too much, it started to overfit the parser, and hence we only trained for 4 epochs.

6.2 Results of the parser approach

The results are displayed in Table 2. As we can see, the results are very good and more or less what we expected. The model managed to shift a lot of the parser's small inaccuracies, the Grammar and Almost mistakes, and make them accurate (In the next section we will show that this is indeed the case). Thus we get an accuracy of 75%, and when including the grammar, which are very small mistakes, we get accuracy of 80.5%, which is state of the art.

6.3 Comparison to the parser

We will now compare the performance of the model against the parser, and see if our prediction, that the model's understanding of natural language will help it overcome the mistakes of the parser, has proven to be true.
Let us look at some example sentences.

1. **Sentence**: What must Link solve throughout the game?
   Model: Link must solve something throughout the game
   Parser: Something must Link solve throughout the game

2. **Sentence**: In what university is the Alexander Graham Bell Institute?
   Model: In a university is the Alexander Graham Bell Institute
   Parser: the Alexander Graham Bell Institute is a university

3. **Sentence**: In 2013, what country did North Korea declare war on?
   Model: North Korea did declare war In 2013 on a country
   Parser: North Korea did declare on someone what country war In 2013

As we can see from the above examples, the model indeed generalizes and corrects the mistakes of the parser. In **example 1**, the parser makes a word order mistake, that makes the premise untrue, but the model corrects it and places the words in the right place. In **example 2**, although both the parser and the model are wrong, the model just shifted the words order a bit, preserving the meaning of the premise, whereas the parser completely failed. We believe that this implies that the model has indeed learned a different way of extracting the premise, one that generalizes better. In **example 3**, the parser generated a jumble of unconnected words which is not even a proper sentence, and the model was able to construct the right premise, which is not even similar to the parser's output.
All of the examples above show that our model did manage to learn a different way of extracting the premise, one that according to the examples and to the results does generalize better than the parser.

7 Conclusion and Future work

7.1 Future work

We believe that our work is the first of many papers that will follow and explore this problem, which was not explored enough so far. We have a lot of ideas on how to improve the model which we did not explore due to lack of time. One of the things we noticed is that many of the mistakes made by the model that was trained on the parser's dataset were simple mistakes of English grammar or a couple

Question Type	True	G - Grammar error	S - Almost True	False
Type 1	70%	10%	20%	0%
Type 2	10%	0%	40%	50%
Type 3	80%	10%	0%	10%
Type 4	70%	10%	10%	10%
Type 5	73%	27%	0%	0%
Type 6	0%	0%	100%	0%
Overall	62.5%	11.25%	16.25%	10%

Table 1: Parser analysis. True means that the premise is exact. Grammar error means it is true except for 1 grammatical error, like "a" instead of "an". Almost True means that the premise is true, except for maybe two adjacent words switching places or another minor change. False is none of the above.

	True	Grammar error	Almost True	False
Model	75%	5.5%	6.5%	13%
Parser	62.5%	11.25%	16.25%	10%

Table 2: Model analysis. The results of the model, and below the parser.

of words that got shuffled in the premise. We think that adding some kind of grammar loss, loss that the model gains when the premise is not grammatically correct, will improve the results drastically.
We tried doing so by fine tuning another copy of BART on a Grammar correction dataset (COLA [4]). We did not change the embedding layer of BART. Therefore, in order to use the premise generation network output as an input to the grammar corrector, we applied softmax on the generated sentence word distributions and aggregated for each word the initial embedding (by matrix multiplication) where the weights are the probability of the word to appear in that place in the sentence (premise generation output). We feed this into the grammar correction network and the loss is now simply the loss of the grammer correction (which is with frozen weights) network.
This implementation did not work, but we think that similar loss may improve the results a lot.

7.2 Conclusion

Our goal in this work was to create a dataset which will enable us to train deep learning models to perform the premise generation task. We think that we indeed achieved said goal, because when training a NLP model on our parser derived dataset, we indeed managed to get a state of the art results. More ever, both of the writers are not linguistics, and from the results of our work we can say with much confidence that using a parser built by expert linguistics will achieve way better results and will generalize even better. This too is a big contribution.

References

[1] Najoung Kim, Ellie Pavlick, Burcu Karagol Ayan, Deepak Ramachandran. Which Linguist Invented the Lightbulb? Presupposition Verification for Question-Answering. 2021

[2] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. 2019

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. 2016

[4] Warstadt, Alex and Singh, Amanpreet and Bowman, Samuel R. Neural Network Acceptability Judgments. 2018