Natural Language Processing HW2

Anonymous ACL submission

Abstract

This document provides details about the second NLP homework assignment. The tasks involve initially fine-tuning and evaluating a large language model (LLM). Following this, I will augment the training data to enhance the model's robustness and improve its performance.

1 Model and dataset

Regarding the model used, I decided to load the pre-trained distilbert-base-uncased model and subsequently fine-tune it with the Fever dataset provided in the homework slides.

1.1 Model and parameters

As mentioned, my homework loads the pre-trained distilbert-base-uncased model, a faster and smaller version of the BERT model, to allow for training and evaluation in reasonable timeframes while still achieving decent results. The parameters used for training are: batch_size = 32, learning_rate = 1e-4, weight_decay = 0.001, and epochs = 1.

1.2 Dataset

Speaking of the dataset, I worked with a subset of FEVER, loaded from HuggingFace using the datasets library. This subset contains samples for training, validation, and testing, with the following features: 'id', 'premise', 'hypothesis', 'label', 'wsd', and 'srl'. The steps I performed to work with this dataset are as follows:

- **Load the dataset**;
- **Map the labels of all subsets (0 = Entailment, 1 = Neutral, 2 = Contradiction)**;
- **Tokenize the dataset**: using the tokenizer provided by distilbert-base-uncased, it assigns a numerical value (token) to each word in the premise and hypothesis, enabling the model to process the data.

2 Data augmentation

Regarding the data augmentation part, utilizing the "wsd" (Word Sense Disambiguation) and "srl" (Semantic Role Labeling) features, along with the assistance of pre-existing NLP models for data processing, I developed and implemented five methodologies, described below.

2.1 Hypernym/Hyponym replacement

The first method I devised involves replacing a word (specially a Noun) in the sentence with its hypernym or hyponym based on a prediction made by the pre-trained BERT model. This is done by masking the target word and predicting the best replacement word within the sentence. Specifically, the steps followed are:

1. Create a list of candidates, obtained from the method that returns the hypernyms and hyponyms of the lemma of the target word.
2. Mask the target word in the sentence, tokenize the sentence with the mask, and process it through the model.
3. Use the model to generate a list of 100 possible tokens to complete the sentence.
4. Calculate the cosine similarity between the generated tokens and all the candidates in the list from step 1.
5. Finally, select the word with the highest similarity to the hypernyms/hyponyms and insert it into the original sentence, converting the word to plural using the inflect library, if necessary.

2.2 Agent and patient swapping

The second technique involves identifying words with the roles of Agent and Patient in the sentences and simply swapping them to generate a sentence that is often false relative to the initial hypothesis.

Nonetheless, a method was created to calculate the similarity between the generated sentence and the original one. Based on a certain threshold, the label of the sentence is appropriately changed.

2.3 Antonym replacement

The third methodology involves swapping a specific word (adverb or adjective) with its antonym (obtained using the method that returns antonyms through WordNet of a lemma), and accordingly changing the assigned label.

2.4 Paraphrasing

The fourth method exclusively utilizes a pre-trained model, specifically Pegasus, tasked with taking the hypothesis from the dataset and paraphrasing it while retaining the meaning. Specifically, the model is asked to return three paraphrased sentences, and through a similarity comparison method, the paraphrase with the highest similarity score to the original sentence is chosen to preserve the same meaning.

2.5 Proper Noun replacement

The last method I conceived involves replacing proper nouns with other proper nouns that are as similar as possible. This technique follows several steps:

1. Gathering lists of proper names from external sources such as GitHub or specific libraries, including names of people (male or female) and cities.

2. Creating a replacement dictionary where the key is the original proper noun in the sentence, and the value is the proper noun with the highest similarity score, obtained by comparing the initial name with the lists gathered in step 1.

3. Implementing a method to substitute proper nouns in the hypotheses of the sentences, with appropriate changes to the label as a result.

3 Reasons of the methods and Data generation

Among the listed methods, I focused more and found more challenging to implement the one related to replacing with hypernyms and hyponyms, as well as the one concerning proper nouns. Specifically for the first method, simply substituting through hypernyms and hyponyms led to personally unsatisfactory results. For this reason, I tried to adapt a strategy that allowed me to draw on these two concepts but ultimately resulted in replacing the word with something rather similar, rather than the exact lemma derived from WordNet. This approach inevitably introduces ambiguity and errors, especially regarding coherence and similarity between the original and the new sentence, as many factors come into play. Nevertheless, my task was also to try to minimize such errors by generating sensible sentences.

Finally, for the automatic and random generation of the new dataset, I initially randomly selected 15,000 samples from the initial training set. Subsequently, I developed a method that randomly chose the function and methodology to apply to each previously selected sample. This approach allowed me to create a new dataset consisting of 15,000 samples generated randomly (plus the other 50,000 from the original set) through augmentation, ready to be used for fine-tuning the model.

4 Results and comparison

Regarding the results obtained, the first fine-tuned model using the original datasets returned: 'eval_accuracy': 0.7022, 'eval_f1': 0.6860. These values, while not excellent, reflect modest architectural choices given the limited computational resources at my disposal. With more precise decisions, such as selecting a better training epoch or a more powerful and efficient model like RoBERTa, significantly better results could have been achieved.

As for the adversarial set, the results are also suboptimal, reported as follows: 'eval_accuracy': 0.5014836795252225, 'eval_f1': 0.5038331002848744.

After performing data augmentation, fine-tuning was painstakingly carried out on the same model using the new dataset. Considering the architecture and parameter choices, the results showed only slight improvement (at least they did not worsen), and are likely much more improvable with more accurate decisions. Specifically, I achieved: 'eval_accuracy': 0.7092260603410582, 'eval_f1': 0.692296233672235. Regarding the adversarial set provided in the slides, I obtained: 'eval_accuracy': 0.5133531157270029, 'eval_f1': 0.5140298545393321.

```
DatasetDict({
    train: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label', 'wsd', 'srl', 'input_ids', 'attention_mask'],
        num_rows: 51086
    })
    validation: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label', 'wsd', 'srl', 'input_ids', 'attention_mask'],
        num_rows: 2288
    })
    test: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label', 'wsd', 'srl', 'input_ids', 'attention_mask'],
        num_rows: 2287
    })
})
```

Figure 1: Original dataset structure.

```
My augmented dataset:
 DatasetDict({
    train: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label'],
        num_rows: 61086
    })
    validation: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label'],
        num_rows: 7288
    })
    test: Dataset({
        features: ['id', 'premise', 'hypothesis', 'label'],
        num_rows: 2287
    })
})
```

Figure 2: Augmented dataset structure.