How Lazy Common Sense Can Be: ComVE Challenge

Chloé Duault, Ellyn Lafontaine, Mallory Taffonneau

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
`{chloe.duault,ellyn.lafontaine,mallory.taffonneau}@fer.hr`

Abstract

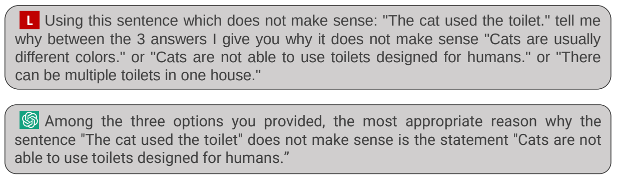
Common sense has been proved to be quite a challenging task in Natural Language Understanding. It requires going beyond pattern recognition. In this paper, we demonstrate how the original BERT model classifier allows us to get acceptable results and how it is possible to fine-tune the BERT model using spacy lemmatization and NLTK's hypernyms and lemmatized synsets. In addition, we provide instances where our model encountered learning limitations and propose ideas of how those issues could be assessed.

1. Introduction

Common sense Validation and Explanation is the task of deciding whether a sentence makes sense or not. It is a hard task in Natural Language Processing since there is no clear definition of common sense, even if it is trivial for humans. Our aim is to determine if a sentence makes sense, and if not why so.

Figure 1 shows a question asked to chatGPT3 and its answer. The question contained a sentence and three options to choose from, to explain why the sentence does not make sense. It is a perfect example to illustrate our goal, and what we are trying to achieve.

To achieve our goal we use a pre-trained BERT classifier. In this paper, we try to find what are the best conditions for BERT to perform this task. Then we will compare the obtained result to ChatGPT3.



L Using this sentence which does not make sense: "The cat used the toilet." tell me why between the 3 answers I give you why it does not make sense "Cats are usually different colors." or "Cats are not able to use toilets designed for humans." or "There can be multiple toilets in one house."

Among the three options you provided, the most appropriate reason why the sentence "The cat used the toilet" does not make sense is the statement "Cats are not able to use toilets designed for humans."

Figure 1: Question asked to ChatGPT3 and answer given back. The sentence and options came from the dataset used.

2. ComVE Challenge

The *ComVE challenge*[1], Commonsense Validation and Explanation, is a challenge from CodaLab Challenge. More precisely, it comes from the SemEval 2020 challenge, we will focus on the fourth task. This task is itself composed of three sub-tasks.

2.1. Description

The first one is to choose between two similar wording sentences one which makes sense and another one which does not. Let's take an example, first we have *"The stove was cleaned with a cleaner."* and *"The stove was cleaned with a mop."*, as you can see there, the problem is not about the grammar, it is only about what does and does not make sense. That is why we need a model that will be able to go beyond pattern recognition.

The second task is to find out why one sentence does not make sense between three possibilities. If we take back our previous example, the sentence that did not made sense was *"The stove was cleaned with a mop"* and for this task, you will need to choose between three possible reasons, first *"The mop was with my mom."*, second *"The mop was my hairy dog"* and last *"A mop is too large to clean a stove"*. Here the correct answer would be *"a mop is too large to clean a stove*", because that is the only answer which explains why you can not clean your stove using a mop. The other two sentences are not explaining why the original false sentence does not make sense, they are just stating random facts that are in no way linked to our original sentence.

The last task consists of generating reasons why one sentence does not make sense. Going back to our example, we could think of *"Mops are one of several tools used for floor care"* as a reason why there is no sense in cleaning the stove using a mop.

In our case, we chose to focus on the first two tasks.

2.2. Our goal

We chose to create one fine-tuned model for each task. Our aim was to be able to determine which features and parameters allowed us to get the best results. Moreover, we also wanted to fathom out what you should absolutely not use in terms of features and parameters.

3. Background

3.1. Dataset

The datasets we used to conduct our experiments were those provided by the challenge (Wang et al., 2020a). They contain a total of 11 997 examples, 10 000 of them are allocated to the training set, 1000 to the testing set and finally 997 to the development set.

We were only able to use 10% of the training set and the whole development set. In Section 4.1. we will explain why.

We split the train dataset as follows : 90% for training and 10% for validation. The validation set has been of great

[1] `https://competitions.codalab.org/competitions/21080`

Table 1: F1 score of the prediction made on our test dataset. It uses a model trained on the train set sub-sampled with different random states.

Random state	55	42	38	22
F1 score	0.50	**0.65**†	0.58	0.54

use to prevent overfitting and to control the learning rate. To test the accuracy of our model, we use the entire development dataset.

3.2. BERT model

The purpose of our work was to determine if a sentence makes sense, and if not, why so. The best way to do this is to use a classifier. We searched for existing classifiers, and we have been able to find a few interesting ones for us: BERT and ELMo. BERT is the best model in our case (Wang et al., 2020b).

We used a pre-trained BERT model classifier provided by the bert-sklearn library Nainan and Medina (2019), which is based on pytorch and transformers to solve our tasks, we used bert cased and uncased during our experiments.

3.3. Features

We did not know from the start which feature would work out the best for our problem. That's why we have decided to try out some of them, we thought could have an impact (either good or bad) on our model. We thought it would be great to use parsing and more precisely semantic parsing.

We first started by the parsing part, using the spacy library to perform lemmatization, stemming, stop word removals, n-gram retrieving. We also used the similarity method from spacy which is computed as the cosine similarity (1) :

$$similarity = \frac{v1.v2}{\|v1\|\|v2\|} \quad (1)$$

Where v1 and v2 are vectors composed of an array of tokens computed between two given sentences. We are using this feature in the second task, when we need to choose between three options for one false sentence.

For the semantic parsing, we used wordnet Yap et al. (2020) with the NLTK library. Wordnet is a large database that contains the semantic relations between words. For a given word, it will create a synset that contains a short explanation of the words, usage examples, it also contains synonyms, hypernyms, hyponyms and meronyms. For us, the important part of wordnet were the synsets and the hypernyms.

4. Realization

4.1. Training

We trained our model using a fine-tuned BERT classifier. We chose to use random state = 42 so that we could have the same results at every training.

Table 1 shows that 42 gave us better results and it is a reference number used in most cases.

Table 2: F1-score for a fine-tuned uncased and cased BERT model on 10% of train set for the first task.

Model	Uncased	Cased
Fine tune BERT	**0.645**	0.568

We encountered a problem while training our model, we did not have the computational requirements to train our model on the entire dataset and for more than 3 epochs. That is why we chose to train our model on only 10% of the dataset and for 3 epochs. We were only able to train using 20% of the dataset one time to get better results. However, we were not able to repeat it multiple times since it required a lot from our computers.

We are perfectly aware that it is an issue and that training on a larger part of the dataset with more epochs could have a huge impact on the quality of our model. But we still think it is great to know, even for a limited part of the dataset what results a user could expect. Since everyone can not have access to an appropriate machine to create its own model.

4.2. Test

Our results have been tested using the F1 score, which is commonly used for classification tasks. We used the F1 binary score for the first task. It was of great use there because the results we expect our model to give for this task are binary, they can only be either true or false.

On the other hand, the second task proved to be more complex and we chose to use the F1 macro score. We chose F1 macro (2) over the F1 micro (3) score because we did not want our model to take unbalanced labels into consideration while computing our score.

$$F1 - Macro = \frac{\sum_{i=1}^{N} F1score_{classei}}{N} \quad (2)$$

With $F1score_{classei}$ the F1 score of the i^{th} class and N the number of classes. Which means, two classes for the first task and three for the second.

$$F1 - Micro = \frac{TP}{TP + \frac{1}{2}.(FP + FN)} \quad (3)$$

With TP, FP, FN as the number of True Positive, False Positive and False Negative.

If only one label was chosen during the prediction, the F1 macro score will show a particularly smaller precision compared to the F1 micro score, which computes the score for the entire set instead of computing the score for each class.

5. Results

The results are obtained with the prediction on our test dataset. The training is realized as explained in the Section 4.1., using the same parameters described.

5.1. First task

For the first task, the obtained results in Table 2 showed that the best case is to use the BERT uncased over BERT cased.

Table 3: F1 score results of the **first Task** with BERT model trained on 10% of the dataset and random state = 42. "No treatment" on the *second row* reference to the model predicting on the **test** dataset without any pre-processing done and "Treatment"in the *third row* referenced to the same pre-processing done on the **test** dataset. "No treatment" in the *second column* referenced to BERT model trained without any pre-processing on the **train** dataset. "Synsets" are lemmatized synsets of each words of the sentence.

F1-Score	Classic	Lemmas	Stemming	Stop Word	Bigram	Trigram	Synsets	Hypernyms
No Treatment	**0.654**†	0.445	0.507	0.648	0.648	0.648	0.636	0.647
Treatment	-	0.528	0.497	0.608	0.608	0.608	**0.649**	**0.649**

Table 4: F1-Macro score results of the **second Task** with BERT model trained on 10% of the dataset and random state = 42. "No treatment" on the *second row* reference to the model predicting on the **test** dataset without any pre-processing done and "Treatment"in the *third row* referenced to the same pre-processing done on the **test** dataset. "Classic" in the *second column* referenced to BERT model trained without any pre-processing on the **train** dataset. "Synsets" are lemmatized synsets of each words of the sentence.

F1-Score	Classic	Lemmas	Stemming	Stop Word	Bigram	Trigram	Synsets	Hypernyms	Similarity
No Treatment	**0.488**	**0.468**	0.278	0.171	0.277	0.182	0.171	0.171	0.178
Treatment	-	**0.492**†	0.300	0.171	0.171	0.171	0.171	0.171	0.171

Following this assumption, all of our experiments for this task have been made using BERT uncased.

When conducting our experiments, we thought about using a multitude of features. In the end, our results show that the best model was the original BERT model, with absolutely no treatment done, either on the train or test set. However, we still found out that hypernyms and lemmatized synsets could show quite some good results.

Furthermore, we discovered that using lemmatization or stemming was a really bad idea, since it gave results close or worse than random.

5.2. Second Task

For the second task, our results show in Table 4 that the best case is to use lemmatization on both the train and test set.

Again, without any treatment on the train and test set we also have good results. In the case of stop word removal, bigram, trigram, synset of lemmas, hypernyms and similarity, the results show that our model is not up to par.

The only time we got some results were for bigram, trigram and similarity without treatment on the test set. In most cases, applying the same treatment and train and test sets are showing an improvement.

5.3. Explanations

In the first task in Table 3, we found out that the best model was BERT without any kind of treatment, which is confirmed by Shi and Lin (2019), closely followed by the synsets of lemmas and the hypernyms. Synsets of lemmas and hypernyms are of great use here, since they are able to feed our model with more information and using this newly added knowledge, our model is better at generalizing. On the other hand, reducing words to simple lemmas or stems has proved to be a really bad idea there.

In the second task, lemmas were one the best results which is in total contradiction with the first task. It led us to think that, in this case, we were able to correctly feed our model in terms of lemmas so that our model could learn efficiently. The classic BERT also obtained some quite good results, as explained by Shi and Lin (2019).

However, we also found out that in the case of hypernyms, synsets, trigram, bigram, stop words removal and similarity our model gave us the same result (e.g. 0.171) over and over again, it is due to the fact that our model failed to learn, each time the model gave us the same answer, (e.g. the label A). We are thinking that for those cases, our model was again underfitted.

As explained before, we were able to train only on 10% of the dataset, since we trained on just a small percentage our model needed to be fed with as much knowledge as possible. All of our experiments tend to show that our model was underfitted to get better results, and we think that some methods used here could have better results than simple BERT when trained on a larger part of the training set.

5.4. Comparison with an other model: ChatGPT3

To compare our results we choose ChatGPT3 since it is a well known state of the art model. We took 50 examples from our test dataset. We then asked chatGPT3 the questions : “which sentence makes sense?” and “Using this sentence which does not make sense: *false sentence*, tell me between the 3 following answers why it does not make sense: option A or option B or option C”, which represent the expectation to the first task then to the second.

The table 5 shows the accuracy obtained on 50 examples for each task and the accuracy of our model. The model is trained with 20% of our dataset for better accuracy. Chat-

Table 5: Accuracy of our model trained on 20% of the data set and ChatGPT3 on 50 examples from our test dataset.

Model	Task A	Task B
ChatGPT3	**0.92**	**0.94**
Fine-tuned BERT	0.71	0.49

GPT3 has better results on the second task compared to the first one, which is not the case of our model, for both tasks accuracy is close to 1. Our model has poor results compared to ChatGPT3, so maybe BERT was not the best pre-trained model to resolve those tasks. However, let's not forget that our model is trained on only a small part of the dataset and on a small number of epochs. Which can explain why we got poorer results than chatGPT3.

6. Related Work

Most of the paper (Shi and Lin, 2019; Yap et al., 2020) we use as a reference focused and tried to create state of art results which was not our case. We mostly focus on finding the best features for each task, but we also want to demonstrate what should not be used in terms of features.

7. Conclusion

In this paper we tried to optimize BERT parameters and input treatment for fitting the Common-sense Validation and Explanation challenge (Wang et al., 2020b). It appeared that the easier option, with no input treatment and most common BERT uncased base model, gave us the best results. Additionally, we compared our results with chat-GPT3, which strongly outperforms our model.

Acknowledgements

This paper was written in the context of a class project for the university of Zagreb. We would like to express our deepest appreciation to Jan Šnajder and Josip Jukić for their advice and guidance throughout this project. Furthermore, this endeavor would not have been possible without Wang et al. (2020b), since they create this challenge. Besides, their paper for the challenge was of great help.

References

Charles Nainan and Ezequiel Medina. 2019. A sklearn wrapper for google's bert model. https://github.com/charles9n/bert-sklearn/.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020a. Github to SemEval-2020 task 4. https://github.com/wangcunxiang/SemEval2020-Task4-Commonsense-Validation-and-Explanation/.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020b. SemEval-2020 task 4: Commonsense validation and explanation.

Boon Peng Yap, Andrew Koh, and Eng Siong Chng. 2020. Adapting bert for word sense disambiguation with gloss selection objective and example sentences.