CS 188 NLP Project Report

Adam Vuilleumier, Brendan Drury,
Gurbir Arora, Ian Conceicao
University of California, Los Angeles
{ianconceicao, bdrury}@g.ucla.edu,
{adamvuille, gurbthebruin}@gmail.com

1 Introduction

In this paper we will discuss how we leveraged the large-scale language corpora pre-trained model BERT and fine-tuned it for the specific text classification downstream task of evaluating whether statements make common sense or not. We trained and evaluated our model on the Com2Sense data set, and used the Sem-Eval-2020 Task 4 for pre-training. In addition, we implemented a masked language modeling pre-train phrase to get the model familiar with the vocabulary. Finally, we leveraged the natural language processing data augmentation library NL-Augmenter in an attempt to further improve model performance.

2 Referenced Related Works

2.1 Com2Sense and Sem-Eval

Com2Sense evaluates a computer's ability to judge whether a given statement makes sense according to its understanding of the world and the semantics of the words in the sentence. To do this, Com2Sense makes predictions on statements that are provided in complementary pairs. In other words, for each statement that makes common sense, there is a second sentence that has been slightly modified which makes it nonsensical to a human interpreter.

In addition, we utilized the Sem-Eval-2020 Task 4 to pre-train the Com2Sense task. Similarly to Com2Sense, the Sem-Eval task evaluates whether an NLP model can correctly classify pairs of correct and incorrect statements. However, the Sem-Eval data set also lists possible reasons why the statements are correct or incorrect (Wang et al., 2020).

2.2 BERT

All of our models use BERT (Bidirectional Encoder Representations from Transformers), a model released by Google AI Language researches in 2018, as a starting point to be fine-tuned. BERT's key innovation is applying transformers to language modeling. Contrary to RNN's, which read left-to-right or right-to-left, a transformer reads an entire sequence of text at once Devlin et al. (2018).

2.3 Masked Language Model

In addition, we also utilized pre-training with a Masked Language Model (MLM). Masked Language Model is when we give BERT a sentence with a few words missing, and train BERT to predict the complete sentence. Further training with an MLM allows us to fine-tune BERT to better understand a particular domain. On top of this, BERT can learn to predict general linguistic patterns and syntax that may be helpful to know Song et al. (2019).

3 Main Methods

3.1 Basic Section

We implemented a checkpoint-best system to record the performance and save the state of our best model in training. In addition, we implemented a way to evaluate performance based on the domains of specific Com2Sense statements. Com2Sense domains label which type of reasoning, between *temporal*, *physical*, and *social*, is necessary to evaluate a statement.

We implemented a grid-search to determine the best parameter values for weight decay, batch size, and learning rate. For each parameter, we chose 2 possible values, leading to 8 total trials. We also adjusted the number of epochs for each trial to fix the total number of optimization steps to be constant across each trial and to ensure fairness. We tested each model on the dev set. See Table 1 for the complete setup.

We ran a masked-language-model as a pre-

Table 1: Grid-Search Paramaters

Trial ID	Weight Decay	Learning Rate	Batch Size
1	0.1	1.00E-06	8
2	0.1	1.00E-06	16
3	0.1	1.00E-05	8
4	0.1	1.00E-05	16
5	0.01	1.00E-06	8
6	0.01	1.00E-06	16
7	0.01	1.00E-05	8
8	0.01	1.00E-05	16

training step to evaluate the potential benefit of a masking step. We also trained a model on the SemEval dataset until convergence and then used it as a starting point for the Com2Sense model to see the potential increase of transfer learning.

3.2 Open Ended Section

In some subfields, data augmentation is utilized to prevent overfitting. For example, in computer vision, images are commonly flipped, rotated or color shifted to normalize the distribution of these features between classes. Data augmentation is less common in natural language processing, owing to the difficulty of creating high-quality synthetic sentences. However, recent work has developed new techniques in this area. Since our dataset is relatively small, with only 800 examples in the training set for com2sense, we decided to expand it with NL-augment, a recently developed data augmentation tool for natural language processing (Dhole et al., 2021). Although this library is intended more for robustness evaluation than expanding training data, as its results are often ungrammatical or have slightly different meaning, the com2sense task is more concerned with sentences' logical content than their exact syntax or connotation.

NL-augment comes with a suite of over fifty different augmentations, but we focused on only five. The "Synonym Substitution" replaces words with synonyms while the "Synonym Insertion" adds synonyms next to them. In theory, this promotes robustness and creates more diversity in sentences, limiting the potential of overfitting. The "Change Named Person" transformation randomly changes any names present in a sentence, weakening the model's ability to try and interpret these names as clues. The "Tense Transformation" randomly changes a sentence into past, present, or future tense by modifying its verbs, again preventing the model from generalizing based on the tense of input sentences. Finally, the "Antonyms Substitute" transformation replaces an even number of words by their antonyms. This double or quadruple negation theoretically cancels out, leaving the logical content of the sentence similar to the original while changing its structure.

4 Main Results

All listed results were evaluated on the development data set unless otherwise specified.

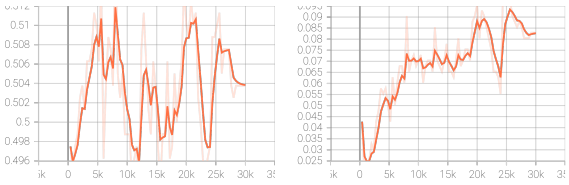
4.1 Grid-Search

Table 2: Grid-Search Results

Trial ID	Accuracy	Pairwise Acc.
1	0.5188	0.1106
2	0.5151	0.1106
3	0.5163	0.1231
4	0.5251	0.2387
5	0.5239	0.0930
6	0.5113	0.0980
7	0.5188	0.1809
8	0.5050	0.1859

As seen above, when all variables were kept the same, but batch size was increased, 3 out of 4 times the model's performance decreased. However, our best model, trial-4 was achieved with a batch size of 16. Similarly, when all variables were kept the same, but learning rate was increased, 3 out of 4 times the model's performance decreased, except for the best trial, trial-4. Lastly, when only the weight decay was increased, half the trials resulted in better performance, and half resulted in worse.

All of these mixed results show selecting the best hyper-parameters is not as easy as increasing or decreasing your variables together because whether a change helps or hurts performance depends on the constant variables. This is why a grid search of every parameter combination is necessary. Our best trial was trial 4, and we used those parameters for the rest of the sections.



(a) Accuracy (b) Pairwise Acc.

Figure 1: Trial-6 Dev Accuracy and Pairwise Accuracy

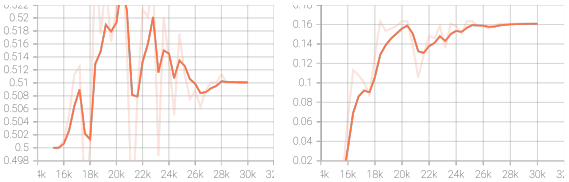
As we can see in figure 1, for trial-6, accuracy began around 0.497, and quickly increased to a maximum at around 8k training iterations. However,

pairwise accuracy continued to increase nearly up until the final epochs. This shows the model has difficulty distinguishing between complementary statements, and that is where most of the growth occurs later in the trial. This makes sense because by purely guessing the same label every time, the model can achieve 0.50 accuracy - the hard part is increasing pairwise accuracy, because this involves predicting opposite labels for each complement in the pair.

4.2 MLM Pretraining

We ran the masking-task for 15k steps and then switched the same model to the Com2Sense task for 15k more steps. We used the same hyper-parameters as trial-4 because it had the best results on the dev set. This brings the total steps to 30k, which was the same as what we did for our grid-search trials, but just broken up now.

The MLM model had a loss of 0.0499, and a perplexity of 1.352. The final model had an accuracy of 0.5330 and a pairwise accuracy of 0.1633.



(a) F1 Score (b) Pairwise Acc.

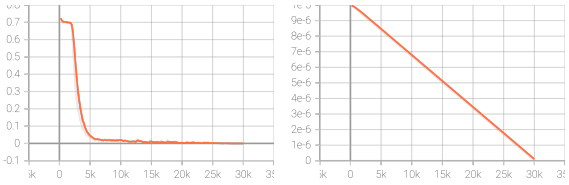
Figure 2: Pre-trained Model Dev F1 Score and Pairwise Acc.

As we can see in figure 2 the F1 Score (which is also the accuracy because of the micro-averaging) started at 0.5, compared to 0.497 without the MLM pre-training, and increased to over 0.52. The pairwise accuracy also quickly increased to 0.16 after only about 5k iterations. The MLM pre-training allowed the model to learn much quicker, and resulted in nearly twice the pairwise accuracy.

4.3 SemEval Knowledge Transfer

We trained a model, again with the trial-4 hyper-parameters, on the SemEval data set for 45k iterations. This model achieved an accuracy of 0.7889. This accuracy was far higher than any accuracy we ever saw on the Com2Sense task for similar training setups, showing the SemEval task is simpler to learn.

Then, we trained it on the Com2Sense task for 30k more iterations. The model achieved an accuracy of 0.5578 and a pairwise accuracy of 0.2437 on the dev set and an accuracy of 0.5366 and a pairwise accuracy of 0.2179 on the hidden test set. These metrics are far higher than our previous best on Trial-4, of 0.5251 and 0.2387, respectively, on the dev set. It shows the SemEval task provided a good foundation for a model to learn Com2Sense on. This makes sense because the tasks are very similar, and the SemEval examples serve as simpler, extra cases for the model to have. Pre-training on SemEval allowed the model to have a stronger foundation and be more generalizable.



(a) Training Loss (b) Learning Rate

Figure 3: Transfer Learning Model Loss and Learning Rate

In figure 3b, we can observe learning rate depreciated as a straight line down to 0 from its initial value. This is the expected behavior from the hugging face get_linear_schedule_with_warmup. The warmup period is not graphed, but during the warmup period the learning rate increase from 0 to the initial value of 10^{-6}.

Figure 3a shows the training loss throughout the training period. Training loss quickly got close to 0 in about 5k steps, but continued to improve until the end.

4.4 Domain Analysis

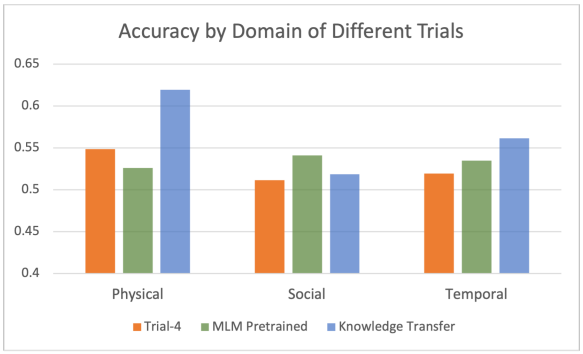


Figure 4: Different Domain Accuracy

To determine how well our models performed on each domain of Com2Sense pairs (ie. physical, social, temporal), we analyzed the accuracy of each model on each of the domains separately (4). Across all the models that were not pre-trained

with MLM, social was consistently the weakest domain. This makes sense as social situations require understanding of human emotions, which are not always logical and are often nuanced. For example, it is easier for a model to understand the concept that some objects are bigger than others but much harder for a model to understand that a researcher might feel envy or even loathing towards a colleague who instantly solved a proof that they had been working on for a long time.

The knowledge transfer model performed the best as a whole on all three domains - particularly physical. Since this model was trained on two data sets, it had much more data to solidify its understanding of the relationships between objects of different physical characteristics.

4.5 Open Ended

Each transformation was applied to each of the almost 800 training examples in the Com2Sense dataset, and the same transformation was applied to both elements of the pair each time. This resulted in an augmented training set with over 3400 examples, as some sentences could not be transformed by specific augmentations and had to be skipped. Models that had been trained on the original com2sense data set were then fine-tuned on this new one, which we named "com2sense-augmented". The augmented sentences were of fairly low quality, as they were frequently grammatically incorrect or unnatural. However, as previously discussed, this was deemed unlikely to be very important for the com2sense task.

Training on augmented data consistently led to improvements on the dev set. In particular, fine-tuning from the best performing model, which used transfer learning from semeval onto com2sense, resulted in a dev set accuracy improvement from 0.558 to 0.567 and pairwise accuracy improvement from 0.24 to 0.25. It is likely that there would have been similar improvements on the test set, as the model was run with the hyper-parameters previously found by grid search and the dev set results were not used whatsoever in the development of the data augmentation technique.

5 Model Checkpoints

Com2Sense Test Set Trial 21
Open-Ended Task

6 Discussion

Looking back on the project, there were a few choices made that we might have done differently given the opportunity to repeat our work:

Firstly, we decided to use micro-averaging to average the scores for each of our trials. After doing this for the first few trials, we considered using binary score averaging - which would have made our recall different from our precision, resulting in more useful evaluation metric - but we decided against it in order to have uniformity across our trials. In hindsight, now that we know that micro-averaging is typically only used for multi-class settings, we probably should have just scrapped our original trials and re-trained with binary averaging. In any case, since the trials were all run with micro-averaging, the results could still be compared to find the best performing model.

Out of curiosity, we evaluated our best model using binary averaging and obtained a precision of 0.553 and a recall of 0.693. This signifies that this model has a low false negative rate - meaning it is biased in favor of predicting correct.

Secondly, although the data augmentation scheme was able to improve the best model even further, various extensions are possible. The NL-Augment library contained additional transformations, many of which used neural methods to achieve complex paraphrases or sentence reordering. Due to computational and time constraints, we did not use these augmentations, but it is likely that they could have led to bigger gains. A custom augmentation tailored specifically to the Com2sense task could likely have yielded the best results. In particular, since many Com2sense examples involve numerical comparisons, we could have written custom logic that created new training examples by tweaking numbers without affecting the relevant comparison.

Lastly, as an extension of this project, we were curious about how these models would perform on Winograd Schemas — a popular alternative to the Turing test. The task is to identify the subject of a verb in a sentence containing a referential ambiguity by using knowledge about the world and the semantics of the sentence. An example of a Winograd Schema is "The trophy couldn't fit in the suitcase because it was too big : What was too big? [Trophy — Suitcase]". It would be interesting to train and evaluate our model on Winograd schemas and how it performs.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Kaustubh D. Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, and Simon Mille. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation. *Preprint*, abs/2112.02721.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. *CoRR*, abs/1905.02450.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. Semeval-2020 task 4: Commonsense validation and explanation. *CoRR*, abs/2007.00236.