CS 188 Project Group 13 Final Report

Aurora Yeh
abyeh@g.ucla.edu

Jason Rauchwerk
jrauchwerk@ucla.edu

Johan Chiang
jchiang87@g.ucla.edu

Dacheng Lin
dlin22@g.ucla.edu

1 Introduction

The goal of this project was to gain experience with the general paradigm of modern NLP research and problem solving. We worked with the provided starter codebase, implementing key components that included loading and processing data, as well as training and evaluation of our models. The two datasets used for this project were the SemEval (Semantic Evaluation) 2020 Task 4 dataset and Com2Sense dataset, both of which contain sensible and nonsensical sentences with corresponding labels. Com2Sense, in particular, provided data in complementary pairs, where both sentences had similar underlying structure, but one made sense while the other did not. We trained models of the BERT family (BERT, RoBERTa, and DeBERTa) to determine the sensibility of a sentence.

Despite the usage of pre-trained models and of techniques such as masking and knowledge transfer, this was an incredibly difficult task. Our best standard accuracy for basic training and evaluation was 0.51703, with a pairwise accuracy of 0.11327. Our best standard accuracy for using knowledge transfer was 0.51935, with a pairwise accuracy of 0.10645. A different set of results for knowledge transfer had a standard accuracy of 0.50573, with a pairwise accuracy of 0.15556.

We also researched and explored ConceptNets to do some knowledge augmentation on the positive labeled sentences of the training data set as a method to improve our accuracy scores. These gave us the incredibly high training and evaluating accuracy of 0.9924 and pairwise accuracy of 0.9974. However, the training process time became longer because of the knowledge augmentation, and maybe we could find better alternative ways to perform augmentation in the future.

2 Main Methods

2.1 Code

com2sense_data.py and **semeval_data.py**

We loaded the Com2Sense or SemEval datasets, depending on which one was specified during the training phase.

Firstly, we read in the data for the Com2Sense dataset. Unique to this dataset is complimentary sentences, each with a unique label (in the training dataset) as to whether the sentence conforms to ”common sense”. Using a loop to traverse all the sentences, we separated the elements that were unique (the sentences and labels), and we combined the rest of the data including domain, scenario, numeracy, and finally we appended them to our dataset. Next, we batch encoded these and created Tensors with the information.

To read in the data for SemEval, we simply loaded the csv file into a `csv.DictReader`. Each row in the SemEval dataset corresponded to a pair of statements. The included reasons were fed into both the correct and incorrect examples (although they are not used in this project).

At this point, we were able to have our model access the training data.

train.py

Firstly, we loaded the configurations of the model we plan to train.

Next, we implemented the training and evaluation loops, computing and saving the loss, and applying softmax to the model's outputs to get the most probable label for the data.

We also computed various metrics: accuracy, precision, recall, f1-score, and pairwise accuracy (for Com2Sense only), which we used to fine tune our hyperparameters.

train_utils.py

We implemented a masking function for pretraining, which replaces most of the input with a mask token and replaces some of the remaining input with a random word, allowing the model to focus on learning specific words or phrases rather than trying to interpret the entire sentence at once.

We also implemented the pairwise accuracy evaluation method, which checks if the pair of corresponding sentences were both evaluated correctly. This is because the Com2Sense dataset provides data in pairs of similarly structured sentences, one with positive sentiment and one with negative.

2.2 Supervised Learning (Com2Sense)

In order to train the Com2Sense model using supervised learning under the `best-based-case` model, we changed the following parameters: `per_gpu_train_batch_size` increased from 4 to 64, `learning_rate` from 1e-5 to 2e-5, `num_train_epochs` from 100 to 150 (although it did not matter since training was stopped around 32 epochs), and `max_seq_length` from 128 to 256. We incorporated a gradient accumulation step size of 2 (up from 1). We also changed the model to log only every 10 steps as opposed to 5 to save on training time.

With regards to the training batch size, we saw from the training metrics with the default parameters that the model was not generalizing quickly enough. We wanted to make use of the GPUs from Google VMs and thus used a batch size of 64. Despite having a faster convergence, the model did not have all the data before it started to learn. Hence, it began making guesses as opposed to learning and making decisions. Furthermore, smaller batch sizes do not guarantee convergence to a global optimum, and our group believed that there was benefits to be gained by seeking guaranteed convergence [2]. By increasing the batch size from 4 to 64, we obtained higher accuracy and pairwise accuracy. Training metrics showed an improvement from 4% to 13% and our pairwise accuracy for the test set was 11%. This demonstrated that the dataset benefited from a larger amount of data before it started to learn.

Next, we looked at the runtime of the training and noticed that it was still taking a while to converge. We decided to increase the learning rate slightly so that we would still have reliable training results without taking tiny steps toward the minimum of the loss function [3]. After comparing the test results of two models, one trained with a learning rate of 2e-5 and one with a learning rate of 3e-5, we determined that 2e-5 yielded metrics higher than 3e-5 (accuracy of 0.517 vs 0.516 respectively).

Finally, we increased the maximum sequence length from 128 to 256 just in case any of the training sentences had a length greater than 128.

2.3 Knowledge Transfer (SemEval)

We hypothesized that a model that has knowledge of the SemEval dataset would perform better than a model only trained with the Com2Sense dataset. We pre-trained different models (BERT, RoBERTa, and DeBERTa) on the SemEval dataset, then fine-tuned the models for a short time with the Com2Sense dataset. We took advantage of the hyperparameter tuning information from the Supervised Learning task.

Our process for the Knowledge Transfer task was to train the models on the SemEval dataset for a relatively long time, then identify earlier iterations which had learned enough. These models were then trained for a relatively short amount of time on the Com2Sense and evaluated. The main hyperparameters we varied were the SemEval-trained checkpoint model we start from, the learning rate of the Com2Sense training, and the Com2Sense-trained checkpoint model we used. Because the SemEval training took so long, we relied on our initial training hyperparameters to be good enough for this task.

2.4 Open-Ended Question: Knowledge Augmentation (ConceptNet)

We chose to use the external source, ConceptNet, to augment some knowledge in order to find a way to potentially improve the performance for the models. ConceptNet is often used to check the semantics and meanings of a particular word and provide related information (eg:other words that has similar context). It is widely used in helping computer programs understand words in more detail. ConceptNet could also help process the statements inside the train.json so more meaningful information could be augmented into these statements that we trained with the models (eg:`bert-large-cased`).

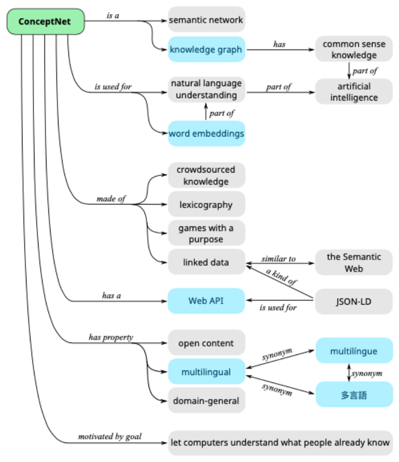


Figure: Picture from ConceptNet Website

First, we sent an API request[1] to the website with the words we tried to mainly focus on (in this case, nouns in the sample sentences). For example, a sentence "Helen loves eating ice cream in hot summer days". The nouns "ice" and "cream" would be used to send our request to gather additional information about the word meanings here. The actual HTTP requests would be "https://conceptnet.io/c/en/ice" and "https://conceptnet.io/c/en/creams", where "en" specifies the model we used was English.

Then, the information we got back from the website was converted to JSON format in Python (ie:the keys for the return dictionary object was "dict_keys(['view', '@context', '@id', 'edges'])"). Here, the "edges" were connecting the words that were related to the current word (nodes in the knowledge graph) to the word we searched (current node). And information could be obtained from the word nodes, such as "ice can cool a drink" and "cream is used for person". With this additional information about the explanation of the nouns, we could embed them into the original sentence; thus, the new sentence after augmentation would be "Helen loves eating ice cream in hot summer days, where ice can cool a drink, [and] cream is used for person."

For this project, we performed augmentation on all the positive sentences in our used to training set. The goal was to improve the training process for the model we used by providing additional knowledge information on the nouns inside the positive sentences.

3 Main Results

3.1 Results for Supervised Learning (Com2Sense)

These parameters trained a model which created a prediction that had the following test metrics (tested using checkpoint 820 and presented in Gradescope Test Trial 22 under Johan Chiang):

- Pairwise Accuracy: 0.11327
- Standard Accuracy: 0.51703
- Standard Precision: 0.51491
- Standard Recall: 0.58781
- Standard F1-Score: 0.54895

The scores are balanced in that we are seeing slightly better results than if our model simply guessed that every sentence had the same label. The recall score indicates that our model was prone to make positive decisions, but the precision indicated that over half of these positives were true. Hence, our F1 score indicates an acceptably performing model.

3.2 Results for Knowledge Transfer (SemEval)

Our results from the Knowledge Transfer task did show improvement from the basic Supervised Learning task. However, our best results only show improvement in either pairwise accuracy or standard accuracy, never both. We found that more training correlated with significantly higher pairwise accuracy, but this same training significantly reduced the standard accuracy.

Our highest standard accuracy result from the Knowledge Transfer task had the following metrics:

- Pairwise Accuracy: 0.10645
- Standard Accuracy: 0.51935
- Standard Precision: 0.51245
- Standard Recall: 0.79642
- Standard F1-Score: 0.62363

This model utilized the `microsoft/deberta-base` base, was trained on the SemEval dataset until checkpoint 4500, and was fine-tuned on the Com2Sense dataset until checkpoint 400. It was submitted to Gradescope Test Trial 19 by Jason Rauchwerk.

An alternative result had the following metrics:

- Pairwise Accuracy: 0.15556
- Standard Accuracy: 0.50573
- Standard Precision: 0.50589
- Standard Recall: 0.49211
- Standard F1-Score: 0.49890

This model utilized the `bert-base-cased` base, was trained on the SemEval dataset until checkpoint 4080, and was fine-tuned on the Com2Sense dataset until checkpoint 3140. It was submitted to Gradescope Test Trial 19 by Aurora Beverley Yeh. (The checkpoint is actually labeled checkpoint-1560, but that was because it used checkpoint-1580 of a previous round of training on the Com2Sense dataset as its base).

3.3 Other Analysis

We believe that the scenario feature would pair best with the domain feature as these two are qualitative features that intrinsically work well with one another. By combining the idea of reasoning and context, we are more likely to obtain better results since the model now learns whether there is a causal scenario that includes a certain category (physical, social, temporal) or if there is a comparison of these states. Although according to the research paper on Com2sense, DeBERTa models perform worse in the physical domain and with a causal scenario [4].

This is an interesting phenomena as our human instincts would have us believe that physical interactions and consequences are clear-cut. However, the limitations of these language models are apparent when comparing the accuracy results by the research team: causal at 62.3% vs comparative at 67.3% [4]. However, casual scenarios are more accurate when taken in the social or temporal domain, and the research team hypothesized that this is due to ”frequent patterns of social activities or sense of time” [4]. Nevertheless, our team posits that a model which has some understanding of these features is more relevant than a model that simply relationships in language.

3.4 Results for Open-Ended Question: Knowledge Augmentation (ConceptNet)

The models we use was `bert-large-cased` for the Com2Sense data set, and the parameters we used for the model were: *per_gpu_train_batch_size=6*, *per_gpu_eval_batch_size=2*, *learning_rate=2e-6*, *per_gpu_train_batch_size=6*, *max_seq_length=6*, *num_train_epochs=100*, *warmup_steps=100*.

Overall, the training and evaluation processes were improved, with the loss decreasing more quickly than expected. The F1-Score increased rapidly because of the additional knowledge for the positive labeled sentence, and the pairwise-accuracy also increased in fewer iterations during the evaluation process.

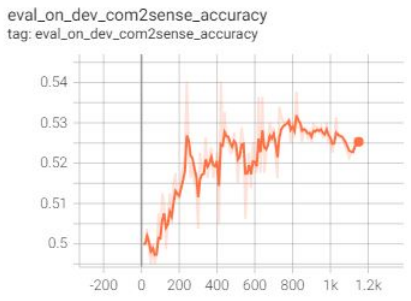
The checkpoint when evaluating with dev.json file that had the best result was checkpoint 320, which had the following metrics:

- Pairwise Accuracy: 0.9974
- Standard Accuracy: 0.9924
- Standard Precision: 0.9924
- Standard Recall: 0.9925
- Standard F1-Score: 0.9924

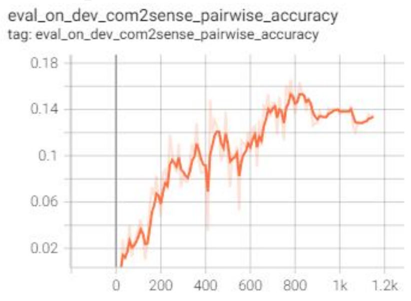
4 Conclusion

4.1 Tensorboard Analysis

Our Tensorflow graph for our Com2Sense training shows large fluctuations in accuracy during the early iterations, indicating that our model may just be guessing.



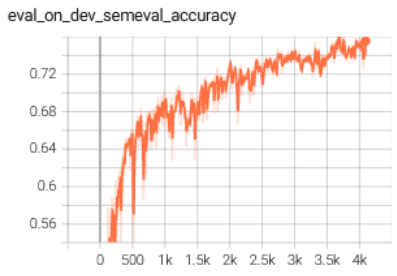
At around step 680, we see the accuracy start to converge and the peak accuracy after this point is at step 820 (which is the checkpoint we used to make the predictions).



Pairwise accuracy follows the same pattern with the peak being at step 820 and then leveling off. Since our model fits well with certain examples and contributes to higher accuracies during those steps, we see some examples not fitting as well which contributes to a higher evaluation loss. However, overall our loss decreases quickly until step 620 where it begins to level off.

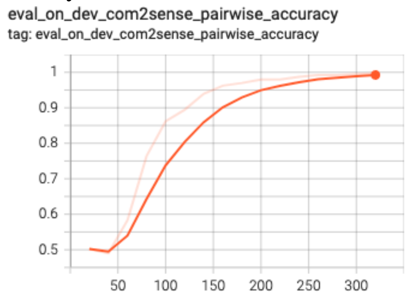
Overall, the predictions for the test set reflect the adequate learning that the model had done. Our model made a prediction for the labels with a 57/43 split. As we know from Com2Sense, a 50/50 split of labels is the true accuracy.

For Knowledge Transfer, we trained on the SemEval dataset for a relatively long amount of time because the accuracy continued to increase.



Afterwards, similarly to what we did for the Com2Sense training, we picked a checkpoint where the accuracy started to converge.

The pair-wise accuracy after Knowledge Augmentation was high, and it tried to converge to 1 after only a few hundred iterations.



At checkpoint 320, it had a almost 100% accuracy because of the additional knowledge augmentation we did for every positive labeled word. Although the time to embed these additional knowledge took longer than expected because of the HTTP requests, the result was converging faster. We suggest that we could try other alternatives, such as importing a word database of ConceptNet, to reduce the access time other the HTTP requesting via the API.

References

[1] "ConceptNet5 Wiki API". In: (2021). URL: https://github.com/commonsense/conceptnet5/wiki/API.

[2] Kevin Shen. *Effect of batch size on training dynamics*. 2018. URL: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

[3] Pavel Surmenok. *Estimating an Optimal Learning Rate For a Deep Neural Network*. 2017. URL: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0.

[4] Te-Lin Wu, Xuezhe Ma, and Nanyun Peng. "Com2sense: A commonsense reasoning benchmark with complementary sentences". In: (2021).