Can text augmentation augment commonsense knowledge?

Marko Čuljak, Darijo Brčina, Vedran Kolka

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{marko.culjak,darijo.brcina,vedran.kolka}@fer.hr

Abstract

Learning commonsense knowledge is a difficult task for machines. However, with the recent advent of transformer models, the ability of NLP systems to learn commonsense knowledge has significantly improved. Motivated by the lack of attention to text augmentation approaches in tackling this challenge, in our research, we experiment with several text augmentation techniques, including a task-specific GPT-2-based technique. We train multiple models in order to explore the effect of text augmentation on the performance of commonsense knowledge validation systems. Our best model yields results comparable to state-of-the-art, with an accuracy of 95.6% on the SemEval-2020 Task 4, Commonsense Validation, subtask A test set.

1. Introduction

Most adult humans are equipped with common sense i.e., according to Cambridge Dictionary, the ability to use good judgment in making decisions and to live in a reasonable and safe way. Even though commonsense knowledge is mostly basic, it is not a fundamental principle of the Universe but rather a collection of knowledge built by generations of humans through observations of phenomena in their surroundings. Since humans are not born with common sense, they start learning it from the earliest stages of their life, either explicitly or implicitly.

On the other hand, teaching computers commonsense knowledge is a difficult task. This is primarily due to two reasons: (1) unlike humans, machines are unable to learn implicitly or passively and (2) commonsense knowledge is broad and constantly evolving. Yet, recently, with the advent of transformer models (Vaswani et al., 2017), the ability of NLP systems to learn commonsense knowledge has improved significantly. The approaches based on fine-tuning transformer models previously pre-trained on large text corpora continuously achieve state-of-the-art results on natural language understanding (NLU) tasks. However, not much attention has been paid to text augmentation, especially in NLU tasks.

In this paper, we tackle the first subtask from SemEval-2020 Task 4, Commonsense Validation and Explanation (ComVE) (Wang et al., 2020) by employing several text augmentation techniques. We explore the effect of these techniques on the performance of common sense validation systems. We experiment with back translation and contextual embeddings-based data augmentation techniques. Furthermore, we propose a text augmentation technique that leverages a GPT-2 model (Radford et al., 2019) to generate artificial data. The subtask on which we evaluate the models' performance requires them to identify which of the given two sentences does not make sense. For example, for the following two sentences:

s_0 : *He poured milk on his cereal.*
s_1 : *He poured orange juice on his cereal.*
the desired output of the model is 1 because the sentence $s1$ is against common sense. The data provided as a part of the task is described in Section 3. The systems, along with text augmentation techniques we experimented with are described in detail in Section 4. and Section 5., respectively. Finally, in Section 6., we present the results of the experiments.

2. Related work

In recent years, many approaches tackle common sense-making and reasoning leveraging transfer learning and transformer models. Wang et al. (2019) present promising results on common sense-making and not as promising results on common sense reasoning. The authors state that the results show that sense-making remains a technical challenge for such models, whereas inference is a key factor that is missing. Nevertheless, the results are meaningful because they show that artificial intelligence systems are still far from human-like performance in common sense reasoning.

The most successful approaches to the Validation subtask (Zhang et al., 2020; Zhao et al., 2020) leverage pre-trained transformer-based models and external commonsense knowledge stored in the ConceptNet (Liu and Singh, 2004). However, as stated by Wang et al. (2020), the success of the ConcepNet-based systems may be attributed to the data leakage which may have occurred because ConceptNet was used as an inspiration for ComVE data generation along with transformer-based models. Rather than leveraging the existing knowledge bases, we attempt to generate additional knowledge artificially by employing text augmentation techniques.

Two approaches at ComVE employed back translation to enhance their common sense validation model performance (Liu et al., 2020; Jon et al., 2020). Although they reported an improvement over their baselines, they do not back their results with statistical significance testing. As for GPT-2 artificial data generation, our approach is inspired by Kumar et al. (2020).

3. Dataset

ComVE subtask A dataset includes 10000 sentence pairs in the training set, 997 samples in the validation set, and 1000

Table 1: The average word counts of sensical and non sensical sentences.

	Training	Validation	Test
Sensical	7.67	7.12	7.25
Nonsensical	7.69	7.16	7.36

samples in the test set. The distribution of labels is balanced. The average word count of sensical and nonsensical sentences in the training, validation, and test set is given in Table 1.

4. Common sense validation model

Our system consists of a pre-trained transformer model as a sentence encoder which is followed by a dropout layer, a linear layer, and a softmax layer. The linear layer is used to map the sentence encoder's output to a single score value. The softmax layer is used as a score comparator. The architecture of the proposed system is visualized in Figure 1.

As encoders we used the following transformer models: DistilBERT$_{\text{BASE}}$ and DistilRoBERTa$_{\text{BASE}}$ (Sanh et al., 2020), ALBERT$_{\text{BASE}}$ (Lan et al., 2020), ELECTRA$_{\text{SMALL}}$ and ELECTRA$_{\text{BASE}}$ (Clark et al., 2020). Due to resource limitations we optimize the hyperparameters only on the model based on ELECTRA$_{\text{SMALL}}$ and reuse the obtained hyperparameters on other transformer-based models. For the same reason we do not consider the largest versions of the aforementioned transformers. Aside from transformers we also experimented with a Bidirectional LSTM (BiLSTM) (Schuster and Paliwal, 1997) as the encoder.

4.1. Implementation details

The transformer models were downloaded through HuggingFace transformers (Wolf et al., 2020) module. Fine-tuning of our model was done using Pytorch Lightning[1] framework in Google Colab[2] environment.

We trained our models using the AdamW optimizer. After a grid search hyperparameter optimization with respect to the provided validation corpus, we concluded the optimal parameters for the AdamW optimizer are 4e-5 for the learning rate, and 1e-2 for the weight decay. We used a linear scheduler with warm-up for adjusting the learning rate during training and we set the number of warm-up steps to correspond to one epoch. Following the same hyperparameter optimization procedure, we concluded that the optimal batch size is 32. As for the dropout layer, we set drop probability for systems trained solely on original data and data augmented by employing back translation and contextual embeddings to 0.1, while for systems that leverage data generated by GPT-2 we set the drop probability to 0.8. We trained our models through 8 epochs, saving the best model with respect to validation loss.

Since the BiLSTM-based model performed poorly on the validation set, we were unable to find hyperparameter values that stand out. We settle for an architecture with 2 BiLSTM layers, a hidden size of 300, a learning rate of 5e-4, and weight decay of 1e-2. As for word representations, we use 300-dimensional GloVe embeddings (Pennington et al., 2014).

[1]https://github.com/PyTorchLightning/pytorch-lightning

[2]https://colab.research.google.com/

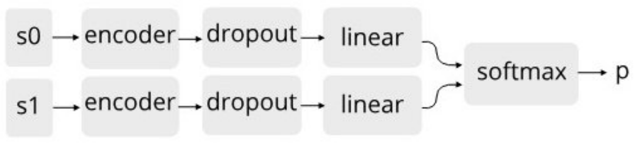

Figure 1: The system architecture used in all experiments. The *encoder* is either a transformer or a BiLSTM and p denotes the predicted probability that $s0$ is against common sense given $s0$ and $s1$ as inputs.

5. Text augmentation

Classic text augmentation techniques are based on deletion, addition, and replacement of certain words as well as changing the word order. Such techniques are not useful for NLU tasks because they may render sensical sentences nonsensical and vice versa. For example, if we change the word order in a sentence *A man eats a cake.* we may end up with a sentence *A cake eats a man.*, which is obviously against common sense. On the contrary, techniques based on alternating single letters in a text are also not particularly useful for NLU tasks since they do not modify the text in a manner significant enough to create new knowledge, although they may contribute to the robustness of a system to spelling errors, which is not essential for such tasks.

However, techniques such as back translation (BT) and replacement of certain words based on contextualized word embeddings (CWE) do not affect the sense of the sentences in a strong manner and may be able to generate additional knowledge, which is why we find them worth exploring. Aside from only modifying the provided data, we attempt to generate artificial sentence pairs by leveraging a GPT-2 model fine-tuned on the provided training data.

We used NLPAug[3] library to perform back translation and CWE-based text augmentation and HuggingFace transformers to obtain the GPT-2 model. After obtaining the artificially generated datasets, we concatenated them to the original corpus, removing the duplicated pairs.

5.1. Back translation

The back translation text augmentation technique leverages two machine translation models: (1) a model that translates text written in language A to language B and (2) a model that translates the obtained text back from language B to language A. The main motivation for back translation is the fact that texts may have multiple valid translations. In this work, we choose German as a language B.

5.2. Contexutalized word embeddings

CWE text augmentation leverages contextualized word embeddings to find a replacement of a word based on its surroundings. The hyperparameters of CWE text augmentation are thoroughly described in NLPAug's documenta-

[3]https://github.com/makcedward/nlpaug

Table 2: Accuracies(%) of GPT-2_{LARGE} with harmonic mean (HM), arithmetic mean (AM), geometric mean (GM), product (Prod) aggregation methods on the validation set. The best two results are in bold.

HM	AM	GM	Prod
65.0	66.4	**74.9**	**78.8**

tion[4]. In this work we use $\text{BERT}_{\text{BASE}}$ for computing the CWE. We modified only one word in each sentence in a pair and set `top_k` to 30 and `temperature` to 0.2. The `top_k` and `temperature` parameters are selected by performing a grid search and manually evaluating the quality of 20 generated sentences.

5.3. GPT-2

Our approach in leveraging GPT-2 for text augmentation can be broken down into four phases: (1) fine-tuning, (2) generation[5], (3) preprocessing, and (4) selection.

Fine-tuning. Firstly, we fine-tune the GPT-2_{MEDIUM} model on the corpus of concatenated sentence pairs obtained from the ComVE training dataset. Each text in the corpus begins with an "S:" string to denote the start of a text, followed by a sensical sentence, "/" token, a nonsensical sentence, and the "<| endoftext |>" token. We fine-tune the model for 3 epochs, with a learning rate of 2e-5 scheduled by the cosine scheduler with warm-up with 300 warm-up steps, and with the batch size of 32.

Generation. In the generation phase we use the fine-tuned model to generate three times as many sentence pairs as the size of the training dataset (30k sentence pairs). For text generation, we use nucleus sampling (Holtzman et al., 2019). This sampling technique randomly selects the next token from the smallest pool of tokens whose cumulative probability exceeds `p`, a hyperparameter which we set to 0.9. To avoid selecting words with extremely low probabilities for the sensical sentence we initially limit the pool size to at most 50 words. After the "/" token is generated we increase the pool size to 100 in order to increase the variability of possible tokens for the nonsensical sentence. The mentioned hyperparameters were selected based on the quality of generated sentences, similarly to the procedure mentioned in Section 5.2.

Preprocessing. The preprocessing phase can be further broken down into 4 steps: (1) elimination of texts that do not match the structure of a valid sentence pair, (2) removing the texts which consist of the same sentences, (3) removing the pairs which are already a part of the ComVE training dataset, and (4) extracting the sentence pairs from the texts. In the first preprocessing step we remove all the

[4]https://nlpaug.readthedocs.io/en/latest/augmenter/word/context_word_embs.html

[5]The code for fine-tuning and generation phases is the adaptation of https://github.com/prakhar21/TextAugmentation-GPT2. Due to resource limitations we were unable to perform hyperparameter optimization for fine-tuning so we used the hyperparameters proposed by the author.

Table 3: Word count means of the first 5k artificially generated sentences in GPT-2-G and GPT-2-P datasets. The average sentence lengths for the GPT-2-G are similar to those of the original training set.

	GPT-2-P	GPT-2-G
Sensical	4.79	8.34
Nonsensical	5.00	7.82

texts that contain three or more sentences, texts that contain sentences with characters other than numbers, letters, and punctuation, or with two or fewer tokens. To tokenize the sentence and obtain the number of tokens we use GPT-2_{LARGE} tokenizer.

Selection. To perform selection of the generated pairs we leverage GPT-2_{LARGE} model. However, in this phase, we use it without fine-tuning. Firstly, we use lm-scorer[6] to assign scores to all generated sentences. The sentence score is calculated by aggregating token probabilities computed by GPT-2_{LARGE}. We consider four aggregation methods: arithmetic mean, geometric mean, product, and harmonic mean. In further research we use only the geometric mean and product, producing two datasets which we denote by GPT-2-G, and GPT-2-P, respectively. The aggregation methods are selected based on the performance of the method on the validation set. To calculate the performance of aggregation methods we firstly calculate the sentence scores using GPT-2_{LARGE} that employs a certain method. We then assign the labels to match the index of a sentence with a lower score. To measure the performance we use the accuracy score. Comparison of the performances is shown in Table 2. The product scoring achieved the best accuracy. This may be attributed to the fact that nonsensical sentences are slightly longer in all the datasets, as shown in Table 1. Since the token probabilities are in the interval $[0, 1]$, their product decreases with respect to sentence length. Therefore, sentences with fewer tokens are scored higher than longer sentences. On the contrary, other methods calculate some form of mean values, which leads to equal treatment of the sentences, regardless of their length.

Finally, we select 10k sentence pairs ranked by the highest difference between the scores assigned to a sensical and a nonsensical sentence. However, we evaluate only the models trained on top 5k artificially generated pairs concatenated to the original data since they achieve the best results on the validation set in comparison to using the top 2.5k or 10k artificially generated pairs. After the selection is performed, we randomly shuffle the sentences in each pair and assign the labels accordingly.

Mean word counts in each of the datasets are shown in Table 3. As expected, due to the nature of the aggregation methods, GPT-2-P contains shorter sentences on average in comparison to GPT-2-G. Figure 2 shows the trends in mean word count in sentence pairs with respect to the number of selected sentences.

[6]https://github.com/simonepri/lm-scorer

Table 4: The accuracies (%) of models based on different encoders trained on the original dataset and the datasets generated by employing different text augmentation techniques. The amount of training examples in each dataset is shown in the brackets. $\#\theta$ denotes the number of parameters of each encoder. In the "Ensemble" column we report the results of the voting ensemble of models in the corresponding row. The best results in each row are in bold.

Model	$\#\theta$	Original (10k)	GPT-2-P (15k)	GPT-2-G (15k)	BT (18k)	CWE (20k)	Ensemble
BiLSTM	3.6M	61.2	63.4	62.5	62.9	61.4	**64.6**
ALBERT	11M	82.8	81.7	80.6	82.4	82.3	**83.8**
DistilBERT	110M	84.8	86.1	85.7	84.0	85.3	**86.4**
DistilRoBERTa	82M	85.7	85.2	85.9	85.6	86.2	**87.0**
ELECTRA$_{\text{SMALL}}$	13M	89.0	89.6	89.4	89.1	89.0	**90.1**
ELECTRA$_{\text{BASE}}$	110M	95.0	93.7	94.6	94.7	94.6	**95.6**

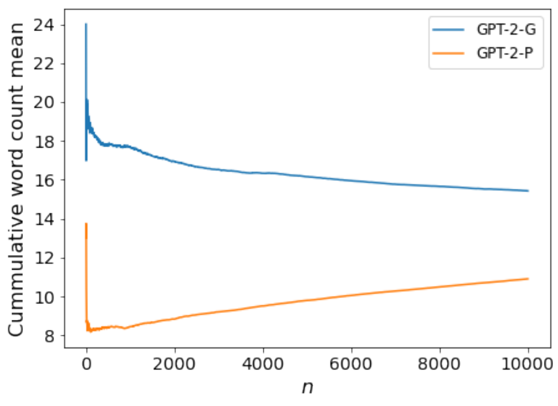

Figure 2: Cumulative means of word counts in top n sentence pairs by with the highest difference between a sensical and a nonsensical sentence in a pair. For example, if n is 100 the point in the graph corresponds to the mean word count in the top 100 pairs.

6. Results

Our main results are presented in Table 4. We evaluated our models on the test set and used the accuracy score as the metric as proposed by the authors of the ComVE task. We refer to models trained on the original dataset as baselines. For statistical significance testing, we employ the one-tailed permutation test with 10000 rounds at $\alpha = 0.05$.

The results show that the models perform similarly when trained on different augmented datasets. There is no notable difference in the performance of the models trained on GPT-2-P and GPT-2-G datasets despite the significant difference in the average sentence length. Nevertheless, the results indicate performance gain when the predictions of the models are aggregated by the voting ensemble. The results of the permutation test show that the ensemble of BiLSTM-based models significantly outperforms the corresponding baseline ($p = 0.045$). However, there is no significant difference in performance between ensembles of models based on ALBERT ($p = 0.246$), DistilBERT ($p = 0.145$) DistilRoBERTa ($p = 0.178$), ELECTRA$_{\text{SMALL}}$ ($p = 0.181$), and ELECTRA$_{\text{BASE}}$ ($p = 0.238$) and their respective baselines. Nevertheless, the performance gain, although not significant when it comes to the transformer-based models, suggests that text augmentation contributes to the diversity in the predictions of the models, which leads to better performance of the ensembles.

Arguably the best results are achieved by models that use ELECTRA as the encoder. ELECTRA$_{\text{SMALL}}$ ensemble significantly outperforms both DistilBERT ($p = 0.005$) and DistilRoBERTa ($p = 0.003$) ensembles despite utilizing notably less parameters. ELECTRA$_{\text{BASE}}$-ensemble performs significantly better than the other models we consider ($p < 10^{-4}$) and its accuracy is comparable to the current state-of-the-art on this task (97.0 %) achieved by Zhang et al. (2020).

7. Conclusion

We show that text augmentation techniques do not significantly affect the performance of transformer-based common sense validation systems. On the contrary, we show that the ensemble of BiLSTM-based models trained on the original dataset and the augmented datasets significantly outperforms the corresponding baseline. Our best model, an ensemble of models based on ELECTRA$_{\text{BASE}}$, achieves 95.6 % accuracy, which is comparable to state-of-the-art.

In future work, we would like to experiment with the proposed GPT-2-based text augmentation technique on less knowledge-intensive tasks. Furthermore, we believe we can improve the proposed technique by utilizing different sampling methods for text generation in order to increase the variability of generated texts. We would also like to experiment with the technique in a low-data regime.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Josef Jon, Martin Fajcik, Martin Docekal, and Pavel Smrz. 2020. BUT-FIT at SemEval-2020 task 4: Multilingual commonsense. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 374–390, Barcelona (online), December. International Committee for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China, December. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

H. Liu and P. Singh. 2004. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October.

Shilei Liu, Yu Guo, BoChao Li, and Feiliang Ren. 2020. LMVE at SemEval-2020 task 4: Commonsense validation and explanation using pretraining language model. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 562–568, Barcelona (online), December. International Committee for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy, July. Association for Computational Linguistics.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. SemEval-2020 task 4: Commonsense validation and explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 307–321, Barcelona (online), December. International Committee for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Yice Zhang, Jiaxuan Lin, Yang Fan, Peng Jin, Yuanchao Liu, and Bingquan Liu. 2020. CN-HIT-IT.NLP at SemEval-2020 task 4: Enhanced language representation with multiple knowledge triples. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 494–500, Barcelona (online), December. International Committee for Computational Linguistics.

Qian Zhao, Siyu Tao, Jie Zhou, Linlin Wang, Xin Lin, and Liang He. 2020. ECNU-SenseMaker at SemEval-2020 task 4: Leveraging heterogeneous knowledge resources for commonsense validation and explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 401–410, Barcelona (online), December. International Committee for Computational Linguistics.