DistilBERT Model with Multi-Scale CNN for Electrical Product Classification

Qian Chen, Haibi Lu
University of California, Berkeley
MIDS, School of Information
W266 Final Project
Instructor: Daniel Cer

Abstract

Market intelligence for electrical product distribution requires sophisticated product classification to capture market share for different product categories, customer segments, and geographic regions. This study focuses on product invoice descriptions provided by manufacturers and retail stores. We performed baseline models, including a simple neural network (NN), a basic Convolutional Neural Network (CNN), and a pre-trained model DistilBERT(a light version of BERT) to extract textual features. After comparing the baseline models, DistilBERT stands out as the best-performing model. We applied multi-scale CNN layers to the DistilBERT model to improve the performance. Our final DistilBERT with multi-scale CNN kernels achieved 0.875 accuracies and an F1-score of 0.889.

Introduction

As the modern electrical product distribution business relies more on market intelligence, one of the biggest challenges that electrical distributors face is categorizing the products from their suppliers to industry standard classifications. As a result, the NAED (National Association of Electrical Distributors) endorses the IDEA (The Industry Data Exchange Association, Inc.) IDW(Industry Data Warehouse) as the source of manufacturer product information.

While the IDW database is a good resource for distributors, they don’t always receive complete product information from their suppliers when processing transactions. The product invoice description is often the only information that comes with the product, and usually, it is just a couple of keywords typed by the salesperson at the counter. As a result, when it comes to analyzing the product classifications, It usually requires an in-house product specialist or a third-party vendor to assign product categories based on the product description, which is not very efficient and inevitably causes a lot of human errors. Due to the number of classifications (over 1000) and the ambiguous descriptions, the human accuracy is estimated at 50-60%. This project aims to develop an NLP model that automates this process, improves efficiency and reduces the number of human errors involved. This can help distributors categorize the products received more accurately and more cost-efficiently for sale or inventory management purposes.

Background and Literature Review

Categorizing product classification based on the invoicing descriptions has two challenges: 1) most descriptions are just keywords that are delimited by comma or similar delimiters (e.g., “Socket Set 13 Pieces 3/4 IN 12 Points”). As a result, they are not typical sequence texts that would work well with RNN and LSTM models. 2）A decent amount of the keywords are numbers (e.g., 3000,4000,5000 are commonly used when describing LED lighting products for their color temperatures). We are not sure how the transformers and encoders would perform with numbers instead of texts.

With this context, we start by looking for a reference focusing on feature extraction. Paulucio Et al.[2] described a state-of-the-art deep neural network model to extract features using BERT and added Random Forest and Neural Network to categorize products based on titles. The study analyzed an extensive e-commerce database with over 20 million datasets and achieved a balanced accuracy of 91.19%. This study used input data from product titles, which gives us confidence that BERT can potentially work well for our feature extraction.

There are several existing studies focused on CNN models for text classification applications. Convolutional Neural Network (CNN) architectures (Kim 2014 [7]) have been used to classify the title of products (Zahavy et al. 2016 [8]). In such a model, the first layer utilizes a random word embedding. Building upon that model, Wirojwatanakul et al 2019 [6] built a modified CNN architecture to classify product descriptions and titles, achieving F1 scores of 77% and 82.7%, respectively.

To address the second challenge, we specifically look for research that focuses on e-commerce product categorizations, which also consist of many numbers in either titles or descriptions. Chen et al [10] demonstrated a multimodal item categorization (MIC) system solely based on a Japanese Text-BERT Transformer. The model achieved 80.2% F1 score over a 500,000-products e-commerce platform. The dataset size is very close to our project.

As we start trying BERT models for our baseline, we quickly realize that it consumes a lot of computation power and is not efficient to fine-tune for our downstream tasks. In search of alternative options, we found a distilBERT(V. Sanh et el 2020[11]) model that is smaller and faster. Z.Liu et el [12] explained a model utilizing a pre-trained distilBERT model and multi-scale CNN for humor detection. It used 5 CNN Kernels on top of the distilBERT model and achieved 94.7% F1-score. Our final model is fine-tuned based on this architecture.

Data

The dataset we use for this project is from an API call to the IDEA connector: https://ideaconnector.idea4industry.com.

To pre-process the data, we first aggregate all the product information provided by the participating manufacturers. (Not all electrical product manufacturers provide product information to IDEA, the IDEA database is estimated to cover ⅓ of the total product market). We remove NAs in the “Invoice Description” and “CMD Category” fields. Finally, we remove duplicated products with the same Invoice Description and UPC number (Product SKUs).
The final version of the dataset contains 572,928 products with 13 features, where the “Invoice Description is our input variable, and the “CMD Category” is our target variable.

There are a total of 2558 different categories. A lot of these categories only have 1 product count. These are usually customized products for specific tasks. If product specialists have never seen the product, it gets assigned to a category by itself. The average length of an invoice description is 28 words. Figure 1 demonstrates the distribution of the input length:

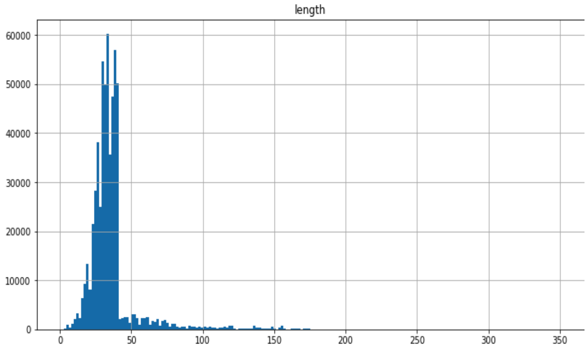


Figure 1: Histogram of Token Length

Methodology and Models

There are 2558 CMD categories, but 1200 have less than 10 products, and 1800 have less than 50 products. We included only classes with a frequency of more than 1000 (determined as “frequency”) in our analysis and dropped those low-frequency products. As a result, 159,837 or 28% of products were dropped. The remaining population contains more than 70% of the initial population, 130 total categories and 413,091 total products.

We believe dropping the low-frequency categories is appropriate due to : 1) In reality, it is appropriate to classify a new product into the existing categories rather than creating a new class, as sometimes there are no substantial distinctions between classes. 2) Alternatively, if we were to assign all these low-frequency products to a placeholder “Other” Category, this would artificially inflate the evaluation score 3) the remaining population (70%) is representative of the entire population and covers the frequent categories.

Baseline Models

We first trained a CNN model as our baseline. We used 80/10/10 split and used Kerastuner to fine-tune our CNN model. Our final model consists of an embedding layer of all the words in the training datasets, a 50 dimensional embedding and the input with a maximum length of 100. We then add one convolutional layer with a dimension of (128,10) and a max-pooling layer. Finally, we added dense layers(output dimension 224) and a dropout layer of 0.5 before the last dense layer with dimension 130. Figure 2 gives the overview of our final CNN baseline architecture.

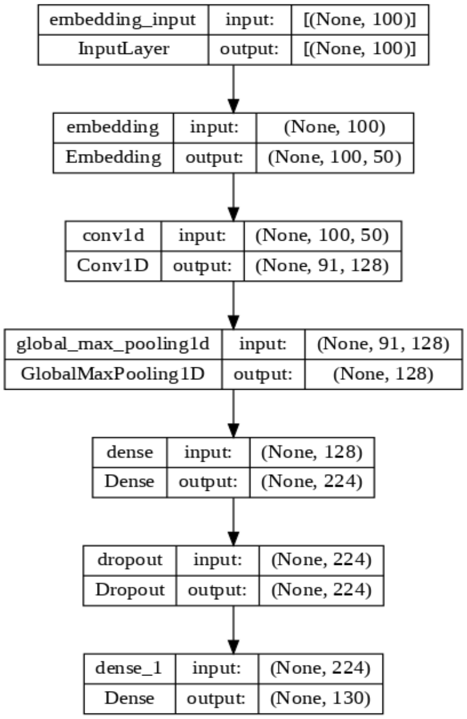

embedding_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding | input: | (None, 100)
Embedding | output: | (None, 100, 50)
conv1d | input: | (None, 100, 50)
Conv1D | output: | (None, 91, 128)
global_max_pooling1d | input: | (None, 91, 128)
GlobalMaxPooling1D | output: | (None, 128)
dense | input: | (None, 128)
Dense | output: | (None, 224)
dropout | input: | (None, 224)
Dropout | output: | (None, 224)
dense_1 | input: | (None, 224)
Dense | output: | (None, 130)

Figure 2: CNN Architecture

The final validation accuracy after running 10 epochs with batch size 32 is 0.7721 and F1 score of 0.792. Figure 3 demonstrates the loss/accuracy over the 10 epochs.

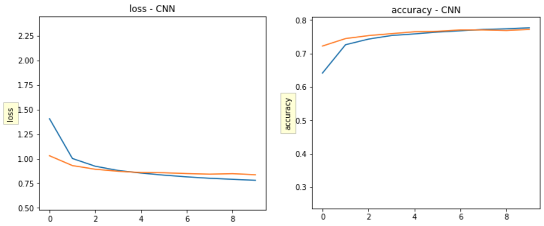


Figure 2: Loss/Accuracy over Epochs for CNN

The second baseline we trained was the DistilBERT model, a smaller, faster, and lighter transformer model trained by distilling the BERT base. It has fewer parameters and runs faster while preserving most of the BERT performance. Fine-tuning the DisBERT model on the entire dataset still requires much computing power. Because of this, in order to run the model faster and have a preliminary sense of the performance of the Distilbert model, we decided to randomly select 10,000 examples from the dataset for the baseline model. We extracted and used the [CLS] token for classification purposes. We then added a dense layer (output dimension 16) and a dropout layer before the last dense layer with a dimension of 130. The final validation accuracy after running 20 epochs with batch size 8 is 0.6557, and the F1-score is 0.6625.

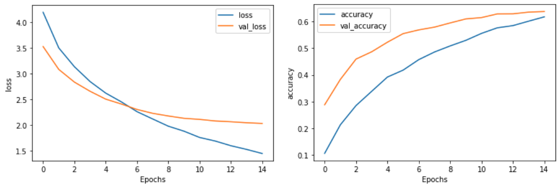


Figure 3: Loss/Accuracy over Epochs for DistilBERT

To compare the DistilBERT and CNN baseline, we also trained the CNN model on the same 10,000 samples and achieved a validation accuracy of 0.600 and an F1-score of 0.632.

Baseline Model	Validation accuracy	F1-Score
CNN	0.600	0.632
DistilBERT	0.657	0.663

Table1: Baseline Model Performance Comparison

Based on the observation from the table above, the accuracy of the DsitilBERT model is around 5 basis points better than that of the CNN model. As a result, we decided to use distilbert as the baseline model and built on top of it for improved results.

DistilBERT+ Multi-Scale CNN

The next step is fine-tuning the DistilBERT model by adding CNN layer(s). We started with just one CNN layer + GlobalMaxPooling1D + dropout and then compared the model by adding multi-scale CNN layers with different filter and kernel sizes. We have tried 3,4,5 CNN layers with increasing kernel sizes and decreasing filter sizes. By training the same 10,000 samples, the three multi-scale CNN layers perform the best.

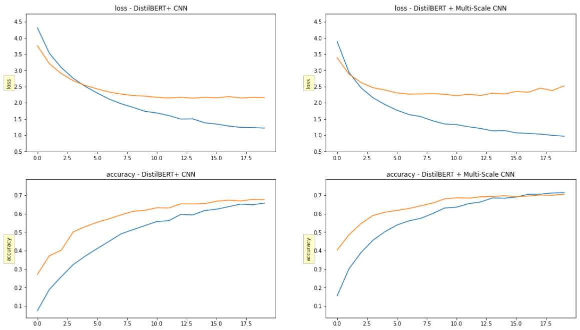


Figure 4: DistilBERT+CNN Vs DistilBERT+ multi-scale CNN

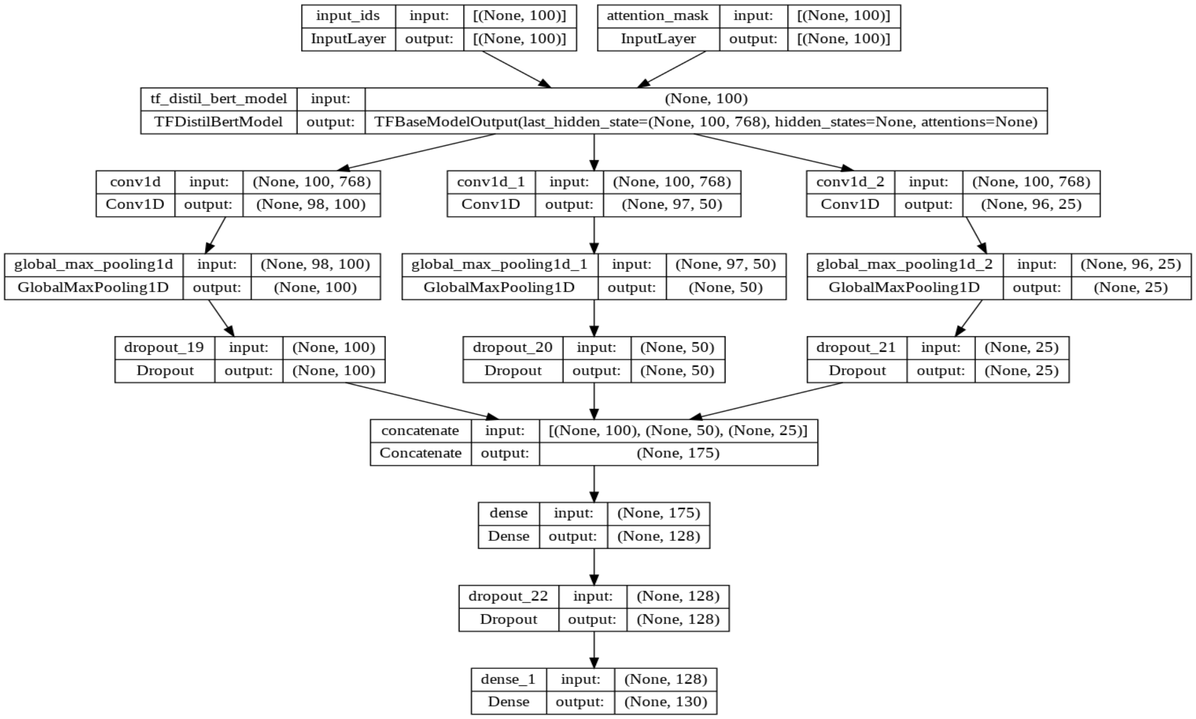

input_ids | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
attention_mask | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
tf_distil_bert_model | input: | (None, 100)
TFDistilBertModel | output: | TFBaseModelOutput(last_hidden_state=(None, 100, 768), hidden_states=None, attentions=None)
conv1d | input: | (None, 100, 768)
Conv1D | output: | (None, 98, 100)
conv1d_1 | input: | (None, 100, 768)
Conv1D | output: | (None, 97, 50)
conv1d_2 | input: | (None, 100, 768)
Conv1D | output: | (None, 96, 25)
global_max_pooling1d | input: | (None, 98, 100)
GlobalMaxPooling1D | output: | (None, 100)
global_max_pooling1d_1 | input: | (None, 97, 50)
GlobalMaxPooling1D | output: | (None, 50)
global_max_pooling1d_2 | input: | (None, 96, 25)
GlobalMaxPooling1D | output: | (None, 25)
dropout_19 | input: | (None, 100)
Dropout | output: | (None, 100)
dropout_20 | input: | (None, 50)
Dropout | output: | (None, 50)
dropout_21 | input: | (None, 25)
Dropout | output: | (None, 25)
concatenate | input: | [(None, 100), (None, 50), (None, 25)]
Concatenate | output: | (None, 175)
dense | input: | (None, 175)
Dense | output: | (None, 128)
dropout_22 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 130)

Figure 5: DistilBERT + Multi-Scale Architecture

We noticed that the loss function presents an interesting pattern where it flattens and then slightly increases over epochs. We think it is due to the incorrect labels generating more "incorrectness" as the training continued. The smaller sample sizes might also contribute to this pattern.

Table 2 compares the accuracy and F1-score between the three models.

Model	Validation Accuracy	F1-Score
DistilBERT	0.657	0.663
DistilBERT+CNN	0.676	0.687
DistilBERT+ Multi-Scale CNN	0.706	0.721

Table 2: DistilBERT Performance Comparison

Final Model

With the confidence that the multi-scale CNN model will perform better, we first trained the same model on the entire dataset. Freeze all layers in the distilBERT model by setting trainable = False. We then fine-tuned the model by unfreezing the base model. One issue we noticed during this process was while it gave us nice incremental improvements, it also led to overfitting.

To overcome this problem, instead of adding the dropout layer after the concatenate layer, we added a dropout layer right after each Conv1D+ GlobalMaxPooling Layer. Our final model consists of DistilBERT features that feed into 3 CNN layers with filters of 100,50,25 and kernel sizes of 3,4,5. Each CNN layer connects to a MaxPooling1D layer and dropout layer (0.5). The concatenated layer feeds into a 128-note FC layer, which finalizes with a 0.3 dropout layer and a 130-class softmax layer. Figure 5 demonstrates the pipeline of our final model, and Figure 6 illustrates the structure of our final model.

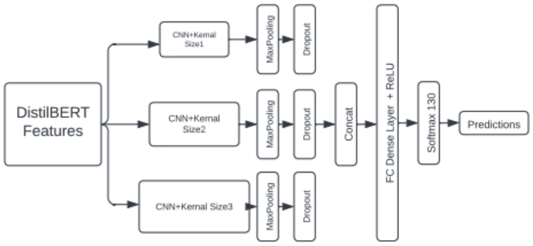
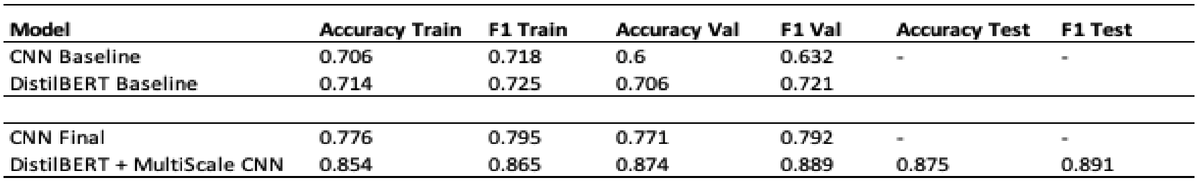


Figure 6: DistilBERT + Multi-Scale Architecture Pipeline



Model	Accuracy Train	F1 Train	Accuracy Val	F1 Val	Accuracy Test	F1 Test
CNN Baseline	0.706	0.718	0.6	0.632	-	-
DistilBERT Baseline	0.714	0.725	0.706	0.721		
CNN Final	0.776	0.795	0.771	0.792	-	-
DistilBERT + MultiScale CNN	0.854	0.865	0.874	0.889	0.875	0.891

Table 3: Evaluation of Final Implemented Models

Results and Conclusions

We trained the final DistilBERT + Multi-Scale CNN model with validation and testing sets. The results are similar. We achieved an accuracy of 0.875 and an F1-score of 0.89. The DistilBERT model performed significantly better than the CNN model we trained initially by 1000 base points. Table 3 above compares both models in detail as well as how they perform against the baseline model. Note that the baseline models were trained with 10,000 samples only, so we can't compare them directly, but in relative scale, it's important to know that the DistilBERT model performs better than the CNN model. It sets the baseline for any future work if this project continues. Figure 7 demonstrates the comparison between CNN and DistilBERT+Multi-Scale CNN model.

CNN *DistilBERT*

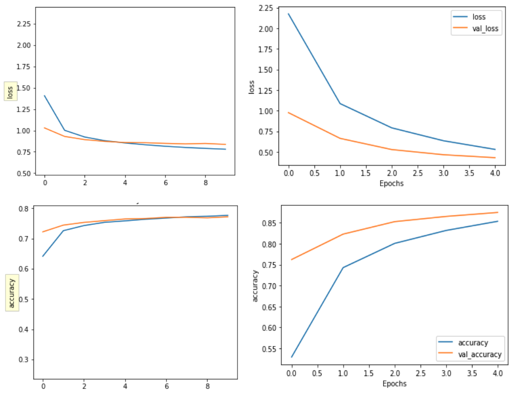


Figure 7: Final Results Comparison CNN: DistilBERT

Recall that initially, we estimate the human accuracy for this task is around 0.5-0.6. This model significantly outperforms manual work in addition to the time-saving aspect.

The increasing loss pattern is less evident in the final (complete) model. Though we have only run five epochs for the DistilBERT model, we have seen converged lines in both loss and accuracy graphs of the CNN model. The increasing loss was primarily due to the smaller sample size; with the complete dataset, it is not as noticeable. If this goes into production, we can compare the AUC-ROC for selected classes and see which class(es) perform worse than the others. Understanding why particular classes perform worse can also help fine-tune the model for future continuation.

Future Work

We considered several things but didn't have the chance to try or implement in this study.

1)With more computing power and time allocation, there are potentially other BERT or transformers we could try to improve the performance further.
2)As we learn more about the BERT base, there are more things we can do to fine-tune the model. In this study, as mentioned in the Methodology section, we unfreeze and freeze the base weights to retrain the parameters and tried different CNN and FC layers to tune the model performance. We could explore the BERT model more by understanding how it was encoded and how we should adapt it to solve the down steam tasks. Merchant el et[14]. is a good reference for illustrating what happens to BERT embeddings when fine-tuning the models.
3)This project only focuses on frequent classes. It would be interesting to split the problem to try different methods on the infrequent classes. We could try the Bayesian inference framework for in-context learning in large language models(Xie el et.[15])

References:

[1] Wangperawong, A. 2018. Attending to mathematical language with transformers. CoRR abs/1812.02825.

[2] L.Paulucio, T.Paixao˜, R.Berriel , A.De Souza, C.Baduea,T.Oliveira-Santos, "Product Categorization by Title Using Deep Neural Networks as Feature Extractor", 2020 International Joint Conference on Neural Networks (IJCNN),28 September 2020

[3] H.Jahanshahi, O.Ozyegin, M.Cevik, B.Bulut, D.Yigit, F.F.Gonen, A.Basar, "Text Classification for

Predicting Multi-level Product Categories", arXiv:2109.01084v1[cs.IR], Sep 2021

[4] W.Yu, Z.Sun, H.Liu, Z.Li, Z.Zheng, "Multi-level Deep Learning based E-commerce Product Categorization", CEUR-WS-Vol-2319, 2018

[5] J,Devlin, M,Chang, K,Lee, K,Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv: 1810.04805, May 2019

[6] P.Wiroj Watanakul, A.Wangperawong, "Multi-Label Product Categorization Using Multi-Modal Fusion Models". arXiv:1907.000420v2 [cs.LG] 17 Sep 2019

[7] Kim, Y. 2014. Convolutional neural networks for sentence classification. CoRR abs/1408.5882.

[8] Zahavy, T.; Magnani, A.; Krishnan, A.; and Mannor, S. 2016. Is a picture worth a thousand words? A deep multi-modal fusion architecture for product classification in e-commerce. CoRR abs/1611.09534.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs]. ArXiv: 1810.04805.

[10] L.Chen , H.Chou , Y.Xia, H.Miyake, "Multimodal Item Categorization Fully Based on Transformers", 2021.ecnlp-1.13, aclanthology.org.

[11] V.Sanh, L.Debut, J.Chaumond, .TWolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", arXiv:1910.01108

[12] Z.Liu, C.Haines, H.Liang,"Utilizing Pre-trained DistilBERT Model and Multi-scale CNN for Humor Detection", UoR at SemEval-2021 Task 7

[13] G. Xiong and K. Yan, "Multi-task sentiment classification model based on DistilBert and multi-scale CNN," 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2021, pp. 700-707, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00117.

[14] A.Merchant,E.Rahimtoroghi,E.Pavlick,I,Tenney, "What Happens To BERT Embeddings During Fine-tuning?",2020.blackbox nlp-1.4

[15] S.Xie, A.Raghunathan, P.Liang, T.Ma, "An Explanation of In-context Learning as Implicit Bayesian Inference",arXiv:2111.02080