THE EFFECTIVENESS OF LARGE LANGUAGE MODELS IN THE MECHANICAL DESIGN DOMAIN

Daniele Grandi, Fabian Riquelme
W266: Natural Language Processing
Fall 2022

ABSTRACT

In this work, we seek to understand the performance of large language models in the mechanical engineering domain. We leverage the semantic data found in the ABC dataset, specifically the assembly names that designers assigned to the overall assemblies, and the individual semantic part names that were assigned to each part. After preprocessing the data we developed two unsupervised tasks to evaluate how different model architectures perform on domain-specific data: a binary sentence-pair classification task and a zero-shot classification task. We achieved a 0.62 accuracy for the binary sentence-pair classification task with a fine-tuned model that focuses on fighting over-fitting: 1) modifying learning rates, 2) dropout values, 3) Sequence Length, and 4) adding a multi-head attention layer. Our model on the zero-shot classification task outperforms the baselines by a wide margin, and achieves a top-1 classification accuracy of 0.386. The results shed some light on the specific failure modes that arise when learning from language in this domain.

1 INTRODUCTION

In the mechanical engineering domain, natural language is used by designers and engineers throughout the design process: to express design requirements, to document design intent, and to communicate ideas and solutions to others in a complex network of people working together to create a single product. One widely used method to document and communicate design decisions is with Computer Aided Design (CAD) software. CAD software allows engineers and designers to create, modify, analyze, and optimize their design, while also documenting various aspects of the design, allowing them to communicate their design choices to others. Using CAD, designers create assemblies of parts, and often use natural language to name each individual part, as well as the assembly itself, for documentation and collaboration purposes, as shown in Figure 1. Often, domain-specific language is used to label parts, which raises questions about the effectiveness of current natural language processing (NLP) techniques in understanding this domain-specific mechanical design language.

NLP techniques have been successfully used to understand the complexity of text and language for multiple tasks such as translation, question answering, and summarization. For these tasks, techniques such as word embeddings, transformers, and attention help navigate the unorganized structure of language. Within NLP techniques, attention-based algorithms have shown state-of-the-art results on NLP tasks as text classification outperforming most featured-based representations methods, e.g., word2vec, Glove, CoVE en ELMo [1]. In this paper, we will explore the use of these techniques in the mechanical engi-

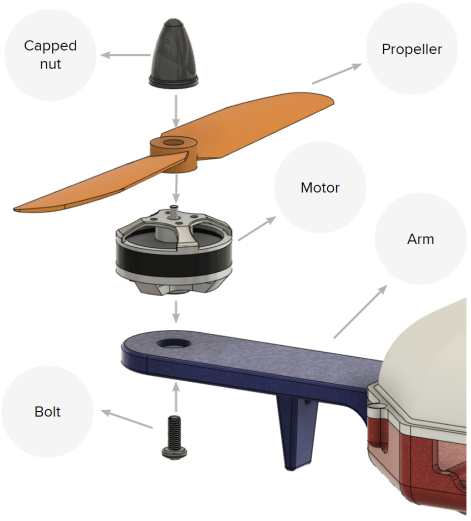


FIGURE 1. An example CAD assembly with each part's semantic name labeled.

neering domain. While some literature has leveraged NLP techniques for data-driven design, little work has been done specifically on natural language used to label CAD assemblies.

To better understand the effectiveness of using Large Language Models (LLMs) in the mechanical engineering domain we will perform an NLP text classification task and recommend changes in the model to improve the results on a domain-specific corpus in the following two steps:

1. We pre-process the ABC dataset to extract and clean a natural language corpus describing the assembly names and part names.
2. We devise two unsupervised tasks to evaluate how different model architectures perform on domain-specific data: a binary sentence-pair classification task and a zero-shot classification task.

The code for this project can be found here[1].

2 BACKGROUND

The motivation of this paper is to further explore NLP techniques in the domain of mechanical engineering and design research. The following is a review of how NLP has been applied in domain-specific applications.

2.1 Usage of natural language data in design research

In recent years, several datasets containing design data have been made available in the research community to support ML research in the mechanical engineering domain. These include ABC [2], the Fusion Gallery Dataset [3], ShapeNet [4], PartNet [5], the Mechanical Components Benchmark [6], FabWave [7], among others. These multi-modal design datasets contain 3D representations of everyday objects (furniture, vehicles, consumer electronics, industrial machinery, etc.) created by engineers and designers using CAD software. However, most of these datasets are limited by the number of classes of objects that they include, they are relatively small compared to 2D datasets such as ImageNet [8], and often they only contain 3D geometry information to represent individual parts (as opposed to assemblies of parts).

Natural language is used in the mechanical engineering domain to assign semantic names to assemblies and their parts. However, of these 3D model datasets, few contain semantic information about the assemblies. One exception is the ABC dataset, which contains 1 million CAD models assembled into about 90,000 assemblies, and also provides semantic names of both the assemblies and the individual parts that make them up. The dataset was scraped from OnShape, an online CAD tool. While the 3D geometry has been used to enable the development of neural networks that learn from 3D data, such as MVCNN [9], PointNet++ [10], UV-Net [11], and BREP-Net [12], fewer work has been done with the semantic names found in 3D model datasets.

Most research done around natural language for data-driven design focuses on sentiment analysis of consumer reviews of products to enable customer requirement prediction [13]. There are some efforts to create and leverage engineering-domain-specific word embeddings: TechNet [14], an engineering domain-specific semantic network, was created by mining semantic relationships of elemental concepts found in US patent data and using Word2vec to create embeddings for 4 million entities. These embeddings have been leveraged for various tasks, such as data-driven design [13], function classification of parts in assemblies [15], and design idea generation [16].

Our work differs from prior literature by focusing on semantic information found in 3D design repositories and by leveraging the implicit knowledge of large language models.

2.2 Applying NLP to new domains

Bert has achieved many amazing results in NLP tasks outperforming other representation methods [1]. Prior to the existence of Transformer architectures such as Bert, there were network-based models (i.e.: recurrent, convolutional) that relied heavily on task-specific neural structures, making them highly complex and computationally expensive. Tasks related to language that require relating signals from two arbitrary inputs or outputs grow in the distance between positions [17] making it even more difficult to explore domain specific tasks as it will require long training on general and specific knowledge, with results that won't compare to the state-of-the-art that Bert has achieved.

Transformers like Bert have been trained on large Text

[1] https://github.com/grndnl/w266_final_project

Book Corpus and Wikipedia articles which give them a broader understanding of language and the contextualized word vectors allow them to understand long-distance dependencies, fine-tuning the transformer on specific data has already shown state-of-the-art performance on widely studied text classification datasets [18]. Therefore, Bert is a good starting point to be applied to new domains with little modification to its Architecture.

2.2.1 Transformer Architecture Bert follows the encoder-decoder framework using stacked multi-head self-attention and fully connected layers [17]. The encoder and decoder contain two sub-layers (a) a multi-head attention layer and (b) a fully connected feed-forward network, while the decoder inserts a third sub-layer which performs multi-head attention over the output of the encoder stack. In both encoder and decoder there are residual connections around each of the sub layers. This structure can be improved on specific task with a process called Bert fine-tuning.

Following previous research on Bert fine-tuning [18], 3 are the main factors at the moment of fine-tuning a Bert Model: 1) Pre-processing the data: focusing on the length of the data, token generation, sentence segmentation and language; 2) Layer Selection, each layer of Bert captures the different features of the input text, selecting the most effective layer for text classification is key, and 3) Overfitting problem and Catastrophic Forgetting: We want to avoid the pre-trained knowledge to be erased during learning of new knowledge.

3 METHODS

3.1 Data Preprocessing

In this work, we leverage the semantic data found in the ABC dataset [2]. Specifically, we use the *assembly names* that designers assigned to the overall assemblies, and the individual semantic *part names* that were assigned to each part.

To extract this useful semantic metadata from ABC, we take the raw data process of each CAD file to extract the *assembly names* and *part names*, yielding text data for 88,886 assemblies. We then preprocess this textual data by deduplicating any assemblies which contain exactly the name *assembly name* and *part names*. Informed with some EDA, we then attempt to clean the text leveraging Python RegEx operations to remove unwanted characters and common strings which were identified to not add any meaningful semantic information. We also perform this deduplication and cleaning step with the *part names*, given that some part names occur very often in a single assembly. This cleaning process results in 61,725 assemblies, 48,644 of which have a unique name.

3.2 Understanding if elements are part of an assembly

To mimic a practical approach to the mechanical engineering world, we are proposing a Bert-based model for a binary Text Classification, constructing auxiliary sentences to fine-tune the model with domain-specific knowledge with the help of specialized layers inspired by Bert's architecture. The prediction of this model will allow us to verify if elements are part of an assembly. The construction of the sentence is considered part of the pre-processing of the data and as mentioned before, this is a critical element to fine-tune a model [19].

3.2.1 Model Architecture Following the Bert architecture the model creation has the following parts:

1. Input Layer: This involves the construction of the sentences for Bert fine Tuning.
2. Bert Encoder: Pre-trained transformer.
3. Output Layer: Attention Multi head layer with a Binary Classification on a Dense Layer on top.

Specifically, given an input text conforming to 2 sentences $X1$ and $X2$, we want the classification model $f(x) = y$ to provide a result $y = 0, 1$ where 0 is 'not related' and 1 is 'related'.

Input Layer In this study we have explored different situations on a base case LLM to understand the basic capabilities of the proposed models. We explored 5 different cases of auxiliary sentence creation, as shown in Table 3 in Appendix A.1, following general structure: "An assembly named 'Name of the Assembly', containing the following parts: "Parts of the assembly separated by comma".

The 5 different cases are tested on a Bert-based model with a binary classification on top.

Base Model and Fine Tuning As a base model we will use a Bert transformer to run a classification task and then use the following strategies to improve the results based on

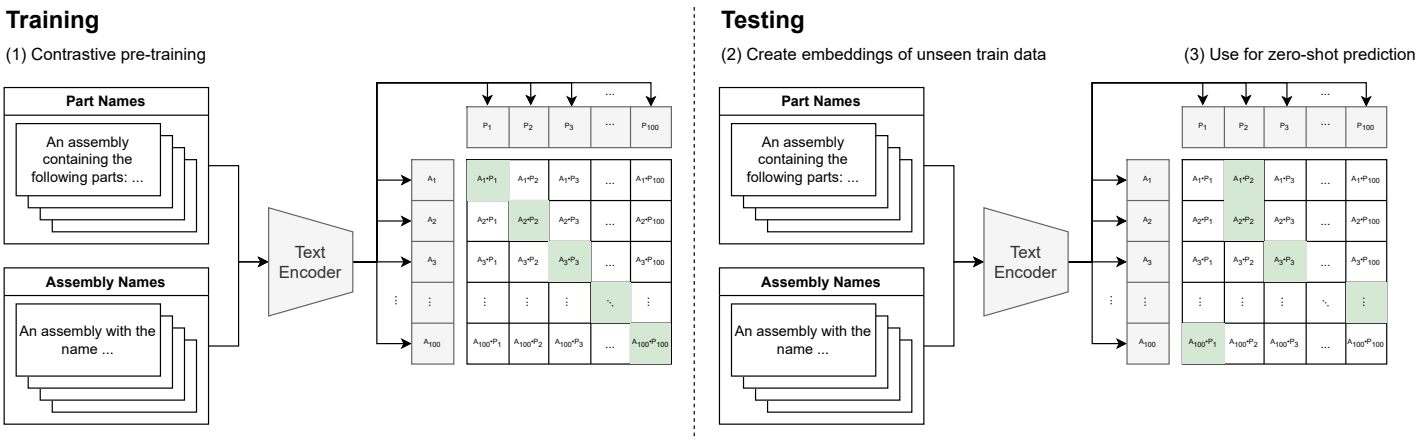


FIGURE 2. Summary of the assembly name prediction task. First, (1) a text encoder is contrastively pre-trained on part name and assembly name pairs. Then, (2) batches of 100 unseen part name and assembly name sentences are encoded using the same text encoder, and (3) the dot product is computed between pairs. The assembly name with the highest value is chosen as the prediction.

the literature review:

a) Learning Rate: Catastrophic forgetting [18] is a common problem in transfer learning, values smaller or equal to 1e-4 have shown good results. For these experiments we will try learning rate values: 1e-2, 1e-3, and 1e-4.

b) Maximum sequence length: We are dealing with small-length sentences so the limitations of Bert model 512 tokens is not a problem. Reducing the size of the sentences could not only increase the testing speed but also improve the accuracy by reducing noise. Most fine-tuning studies focus on sentences containing more than 512 tokens [1]. In this study, we will evaluate the results with 128 tokens and 256 tokens on the base BERT model.

c) Number of multi-attention heads: As multi-head attention is a driving force behind Bert Architecture [19], changing the amount of it could improve the results. Base Bert model has 8 Multi-head attentions on the transformer, adding a layer of multi-head attentions. Initial indications show the model improves with more heads, in this study we will test with 8 and 32 multi-heads on the attention layer.

d) Trainable Last Layer: Each layer of Bert captures the different features of the input text [18] have shown that fine-tuning the last layer of BERT gives the best performance for a text classification task. In this study, we will use this recommendation while fine-tuning the model.

e) Dropout probability: Initial experiments show that the amount of unique data is prone to over-fitting. El [20] proposed that tuning the dropout hyper-parameter on a dropout layer leads to different model performance. In this study, we will experiment with a dropout of 0, 0.1, and 0.3.

3.2.2 Evaluation For these experiments we will evaluate Training accuracy and Validation accuracy as we are most interested in the amount of correct answers on the binary classification.

3.3 Assembly name prediction

The second task we formulate is an assembly name classification task. As Section 3.1 describes, after preprocessing there still remain 48,644 unique assembly names in our corpus. To simplify the multi-class classification task to a more reasonable 100 classes, we evaluate our model by using the text encoder as a zero-shot linear classifier to predict the assembly name of each set of 100 part names per batch, borrowing the idea from CLIP [21].

3.3.1 Model architecture As shown in Figure 2, we train the model leveraging a contrastive approach (1) where we use the same text encoder to generate embeddings for the *assembly names* as well as the *part names*. The network is trained to create embeddings for both sets of names, such that each *assembly name* and its corresponding *part name* embeddings are located near each other. Then, (2) during testing, we batch the test data into batches of 100, and generate embeddings for each *assembly name* and *part names* string. The order of the *part names* embeddings is then randomly shuffled, and (3) we treat each batch as a 100-

TABLE 1. Bert fine tuning results for different hyper-parameters and architectures (train/test accuracy).

Models	Learning rates	No Dropout	Dropout 0.1		Dropout 0.3	
		Seq. Len=256	Seq. Len=128	Seq. Len=256	Seq. Len=128	Seq. Len=256
Base Bert	lr=0.001	0.546/0.621	0.537/0.614	0.542/0.699	0.547/0.592	0.526/0.505
Base bert Model 8 heads	lr=0.01	0.565/0.515	0.529/0.519	0.585/0.527	0.526/0.509	0.544/0.605
	lr=0.001	0.722/0.627	0.693/0.603	0.720/0.569	0.691/0.571	0.724/0.543
	lr=0.0001	0.696/0.551	0.683/0.529	0.718/0.611	0.687/0.594	0.710/0.501
Base bert with 32 attention heads	lr=0.01	0.555/0.516	0.563/0.533	0.578/0.539	0.572/0.468	0.527/0.602
	lr=0.001	0.720/0.560	0.610/0.582	**0.723/0.618**	0.697/0.615	0.723/0.602
	lr=0.0001	0.722/0.598	0.680/0.591	0.710/0.531	0.683/0.520	0.712/0.520

class zero-shot classification task by normalizing the embeddings, computing the dot product between the *assembly name* and *part names* embeddings, and selecting as the prediction the pairing with the highest value. We repeat the last step for each batch in the test data, and average the results.

Following the fine-tuning strategies outlined in Section 3.2.2, we perform a hyperparameter search by varying the number of frozen Bert layers, dropout amount, learning rate, temperature value (a parameter of the contrastive loss calculation), and Bert output token. We also try different text encoder architectures by adding additional attention layers, or additional fully-connected layers on top of Bert.

3.3.2 Evaluation For this task, we chose accuracy as the evaluation metric, as each batch of 100 test classes differs from each other, it is not meaningful to average the precision and recall results of each batch.

We evaluate our model against two baselines: a pre-trained Bert base uncased model, and a Bert base uncased model that we fine-tune on the training data on a masked language model (MLM) task.

We take these models and use them as text encoders in the testing pipeline shown in Figure 2 to create embeddings of the test data and compute the zero-shot prediction accuracy.

4 Results and Discussion

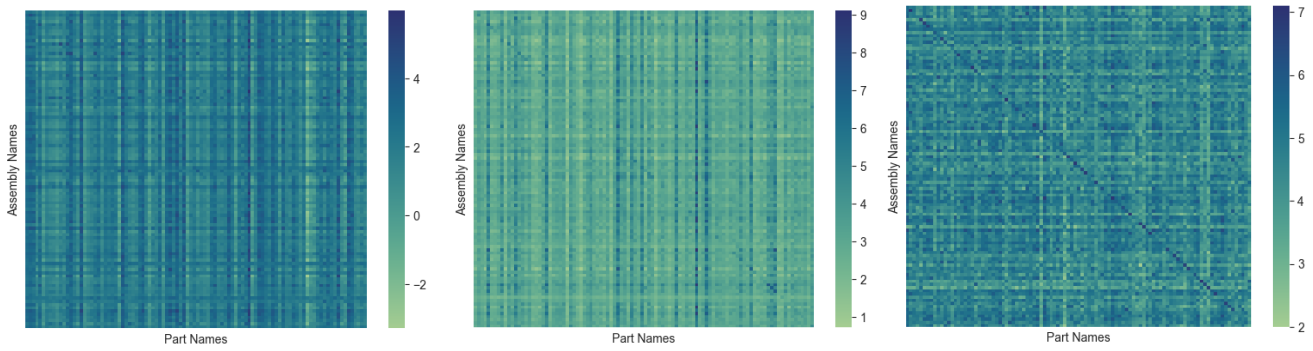
4.1 Entailment task

In this section, we discuss the results of the Binary classification of auxiliary sentences of mechanical element part.

4.1.1 Quantitative With the help of the auxiliary sentence creation, we evaluate the correct prediction of the second sentence to be the continuation of the first one. Incorrect pairings were created at random with the data from the same data set. The results of this experiment can be seen in Table 1.

The experiment reveals the following: **Learning rate**: We see an increase of both Validation Accuracy and Test accuracy as Learning Rates increase. The best results were obtained with 0.001 learning rate which was not the smallest and differs from Sun's et. al. recommendation [18]. **Dropout**: The main tendency we observe is the increase the test accuracy but there was not significant change with the train accuracy. **Sequence Length**: Sequence length of 256 have better results than 128. This is expected as with 128 we are removing significant information of the sentences that can be used by the model to understand relations.

4.1.2 Qualitative Our evaluation shows two limiting factors: the scarce amount of data, and the large number of unique values on the dataset. Overfitting was a common problem during training. We observed that the results of learning rates differ from the recommendations of the literature are we are dealing with a more unique dataset, overfitting the model is easier with more epochs, and learning rates need to increase along with dropout to contrast this effect. Finally, an attention layer with multiple heads is beneficial to the model. This is understandable as each head will focus on a different aspect of the model at the same time, each time it will evaluate the importance of the model reducing the risk of over-fitting and at the same time encounter long-distance relations on the sentences which will help with the uniqueness of the data-set.



(a) (b) (c)

FIGURE 3. Cosine similarity between the normalized embeddings of the first batch of 100 assembly names vs. 100 part names for (a) Bert base, (b) Bert fine-tuned, and (c) Bert pretrained. A 100% correct result would yield a figure where the items on the diagonal have the highest cosine similarity, indicating that the embedding of the assembly name and the embedding of the part names are most similar.

4.2 Assembly Name Prediction Task

In this section, we discuss the results of the assembly name prediction task.

4.2.1 Quantitative We formulate the assembly name prediction task as a 100-class classification task. To evaluate the performance of different text encoders, we calculate the mean accuracy on the 12,321 unseen examples in the test data by taking the average accuracy across 123 batches of 100 examples.

TABLE 2. The results of the assembly name prediction task.

Method	Mean Test Accuracy		
	Top-1	Top-5	Top-10
Random Chance	0.01	0.05	0.10
Bert	0.058	0.158	0.240
Bert + Fine-tuning	0.095	0.210	0.303
Bert + Constrastive Pretraining	**0.386**	**0.553**	**0.632**

Table 2 shows the top-1, top-5, and top-10 mean test accuracies of the three text encoders: the Bert base uncased model, the fine-tuned Bert Base uncased model, and our pretrained Bert model using a contrastive approach. Our model outperforms the baselines by a wide margin, and achieves a top-1 classification accuracy of 0.386.

4.2.2 Qualitative The results can be visually inspected by plotting the cosine similarity between the assembly name and the part name embeddings generated by each text encoder. Figure 3 shows a batch of 100 assembly names by 100 part names in the test set. Our model shows a very distinct pattern compared to the baselines. While the baselines (a and b) showcase an interesting pattern where for each part name embedding, all assembly name embeddings are either high or low (seen as vertical lines of either high or low values), our model (c) does not showcase this pattern. In Figure 3(c), our model can be seen to only assign a high cosine similarity value to a few assembly names for each part name embedding, which, in line with the qualitative results, seems to result in a higher overall accuracy. Also to note, is the different scales of each subplot: the better-performing models create more distance between correct and incorrect results.

To further evaluate our results, we can look at some predictions done by our model, as shown in Table A.2 in Appendix A.2. From the list of correct predictions, we can see that the model is very good at picking the correct assembly name when the number of parts is low (one or two). The task becomes very simple when the part name is exactly or in part the same as the assembly name; we expect the baselines to also pick up on this. For example, an assembly that contains the parts 'lens, adapter' is correctly classified with the name 'macrolens adapter for nexus 5x'.

We can also categorize the failure modes as follows. Some assemblies do not contain natural language, but rather part IDs, which we assume the model has never seen during pretraining. Some assembly names do not contain

meaningful semantic information, such as 'untitled document' or 'liams project', which makes it very hard for the model to infer any connection with the list of part names. Different sets of assembly names could contain similarly named parts, such as 'bed frame' (ground truth) and 'picnic table' (predicted). Overall, the qualitative evaluation showcases how challenging this task might be even for a human, and a human baseline study would likely shed more light on the failure modes.

4.3 Limitations and Future Work

The fine-tuned model developed in this study could be used for the Assembly Name prediction task as an encoder. The limited data available could be augmented to get more training and testing sets. In the future, we aim to create generative models trained on this type of data that are used to create a list of parts given an assembly name and a description of the function of the assembly, as a starting point for the mechanical designer.

5 Conclusion

Natural language is used by mechanical engineers to assign semantic names to both assemblies and their constituent parts during the design process, for the purposes of documentation and collaboration. This work explored the effectiveness of using large language models to understand the jargon used by designers in the mechanical engineering domain.

To this end, we devised two unsupervised tasks: a binary sentence-pair classification task and a zero-shot classification task, which allowed us to evaluate various architectures for capturing domain-specific language. We achieved a 0.62 accuracy for the binary sentence-pair classification task, and 0.39 top-1 accuracy for the 100-class zero-shot assembly name classification task. The results shed some light on the specific failure modes that arise when learning from language in this domain. Future work could leverage the tasks we devised to assess both the quality of other design corpora or search for better model architectures to understand language in the mechanical engineering domain.

REFERENCES

[1] Yu, S., Su, J., and Luo, D., 2019. "Improving bert-based text classification with auxiliary sentence and domain knowledge". *IEEE Access,* **7**, pp. 176600–176612.

[2] Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D., 2019. "Abc: A big cad model dataset for geometric deep learning". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9601–9611.

[3] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., 2015. "3d shapenets: A deep representation for volumetric shapes". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920.

[4] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al., 2015. "Shapenet: An information-rich 3d model repository". *arXiv preprint arXiv:1512.03012*.

[5] Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., and Su, H., 2019. "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding". In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 909–918.

[6] Kim, S., Chi, H.-g., Hu, X., Huang, Q., and Ramani, K., 2020. "A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks". In European Conference on Computer Vision, Springer, pp. 175–191.

[7] Bharadwaj, A., Xu, Y., Angrish, A., Chen, Y., and Starly, B., 2019. "Development of a pilot manufacturing cyberinfrastructure with an information rich mechanical cad 3d model repository". In International Manufacturing Science and Engineering Conference, Vol. 58745, American Society of Mechanical Engineers, p. V001T02A035.

[8] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., 2009. "Imagenet: A large-scale hierarchical image database". In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255.

[9] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller,

E. G., 2015. "Multi-view convolutional neural networks for 3d shape recognition". In Proc. ICCV.

[10] Qi, C. R., Yi, L., Su, H., and Guibas, L. J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

[11] Jayaraman, P. K., Sanghi, A., Lambourne, J. G., Willis, K. D., Davies, T., Shayani, H., and Morris, N., 2021. "Uv-net: Learning from boundary representations". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11703–11712.

[12] Lambourne, J. G., Willis, K. D. D., Jayaraman, P. K., Sanghi, A., Meltzer, P., and Shayani, H., 2021. Brepnet: A topological message passing system for solid models.

[13] Feng, Y., Zhao, Y., Zheng, H., Li, Z., and Tan, J., 2020. "Data-driven product design toward intelligent manufacturing: A review". *International Journal of Advanced Robotic Systems,* ***17***(2), pp. 1–18.

[14] Sarica, S., Luo, J., and Wood, K. L., 2020. "Technet: Technology semantic network based on patent data". *Expert Systems with Applications,* ***142***, p. 112995.

[15] Ferrero, V., DuPont, B., Hassani, K., and Grandi, D., 2022. "Classifying component function in product assemblies with graph neural networks". *Journal of Mechanical Design,* ***144***(2).

[16] Sarica, S., Song, B., Luo, J., and Wood, K. L., 2021. "Idea generation with technology semantic network". *AI EDAM,* ***35***(3), pp. 265–283.

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., 2017. "Attention is all you need". *Advances in neural information processing systems,* ***30***.

[18] Sun, C., Qiu, X., Xu, Y., and Huang, X., 2019. "How to fine-tune bert for text classification?". In China national conference on Chinese computational linguistics, Springer, pp. 194–206.

[19] Michel, P., Levy, O., and Neubig, G., 2019. "Are sixteen heads really better than one?". *Advances in neural information processing systems,* ***32***.

[20] El Anigri, S., Himmi, M. M., and Mahmoudi, A., 2021. "How bert's dropout fine-tuning affects text classification?". In International Conference on Business Intelligence, Springer, pp. 130–139.

[21] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I., 2021. Learning transferable visual models from natural language supervision.

A Appendices

A.1 Task 1: Sentence Formation

TABLE 3.

Identification	Sentence 1	Sentence 2	Accuracy
Base Case	Name of Assembly	Name of Parts	0.469
Case1	Name of Assembly + Part1	Name of Part 2 to the end	0.549
Case2	Name of Assembly + Parts minus last one	Name of the last part	0.557
Case 3	Name of Assembly + half name of parts	Name of the second half	0.630
Case 4	Name of Assembly + Description	Name of parts	0.594

A.2 Task 2: Assembly Name Predictions

TABLE 4. A list of correct predictions and incorrect predictions done by our model.

Correct Results		**Incorrect Results**		
Part Names	Assembly Name Ground Truth	Part Names	Assembly Name Ground Truth	Assembly Name Prediction
20mm stack	20mm stack	end 1, long divider, end 2, side 1, short divider, side 2	box shell version	ufo
2238 375	2238 375.step	mainbox, boxlid, internalcomb, basebottomkeyotherhalf, windholder, barform, basebottomkeyhalf, basetopkey	ww sport base bar box	bed frame
cap, sleeve	sleeve	171, motor shaft, vt3m4x12, rail cross part, bed rotor mount, 168, 151, base board, 170, 163, 176, build plate prt11, 182, z scr	circle builder	300 x 20 stainless chuck
top line holder, bottom line holder, rod carrying tube, rod shaft	line holder	sproket, spring link, bearing, stopper, shaft, spring arm, arm support	untitled document	box shell version
8 andr\\x2\\00e9\\x0\\ mouri\\x2\\00f1\\x0\\o fig11 78	11.78	motherboard panel, lower front plate, motherboard dock panel, punched angle bracket, hex bolt 3 8 1l 93190a624, hex bolt 3 8 1 2	crop	drip disk 2
tapa limaton opuesto 2, ubicaci\\x2\\00f3\\x0\\n villa luz google maps y levantamiento topografico 2, perfile limaton 2, correa	ubicación guayacanes hermanos	bras sup acier1piece	ierlgihrlihjcehveobv.step	schlüsselbrett
rod glue jig, v slot addon, big joint jig, rod cut jig	rod jigs	4 laakerinen uusi malli mago	spinner mago 4 laakerinen	300 x 20 stainless chuck
piggy bank plug	piggy bank plug	mxf, mpw, mtms, button, connector, pcb, intercom housing, mxa, front panel, keypad, standoff short, standoff long, spacer, 13 pi	intercom keypad controller	cap smd 2220.step
roller4 1	roller 1	index mold bottom, app index, ring mold bottom, ring mold top, ring, thumb proto base, index mold top.	finger	various 3d prints
lamphoudertje	houder bedlampje ikea	main support, support, ns inside panel, ns front panel, end, ns back panel, ns outside panel, side, ns bottom panel	bed frame	picnic table
heton ep	het	user library smd capacitor	cap smd 2220.step	pc ie mini card connector 1.5mm gap
804 9303	804 9303	clamp, brake, grip, handle, wheel, fork	liams project	ufo
13mm nut, first part spanner	first part spanner	rubber foot, foot stud, foot nut	co mac 3894 swivel foot	roller 1
lens, adapter	macrolens adapter for nexus 5x	case, inside	cnc bit holder	line holder