Patentformer: A Novel Method to Automate the Generation of Patent Applications

Juanyan Wang[1]
juanyan.wang@partner.samsung.com

Sai Krishna Reddy Mudhiganti[1]
s.mudhiganti@samsung.com

Manali Sharma[1]
manali.s@samsung.com

[1]Samsung Semiconductor, Inc.
San Jose, CA

Abstract

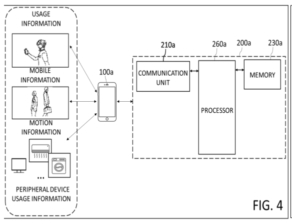
In recent years, Large Language Models (LLMs) have demonstrated impressive performances across various NLP tasks. However, their potential for automating the task of writing patent documents remains relatively unexplored. To address this gap, in this work, we propose a novel method, Patentformer, for generating patent specification by fine-tuning the generative models with diverse sources of information, e.g., patent claims, drawing text, and brief descriptions of the drawings. To enhance the generative models' comprehension of the complex task of writing patent specification, we introduce a new task, *claim+drawing-to-specification*, and release a new dataset. We evaluate our proposed method on thousands of patents from the USPTO[1] and show that our method can generate human-like patent specification in legal writing style. Human evaluations by four patent experts further affirm that our proposed method has the potential to generate correct specification, and the quality of generated specification may sometimes be better than the actual specification.

1 Introduction

Patents are legal documents that require a very specific writing style where certain words and phrases carry specific meanings, e.g., an "embodiment" of the invention refers to the physical manifestation of the invention or idea. A patent document usually consists of the title, abstract, field of the invention, background, summary of the invention, independent claims, dependent claims, drawings, brief descriptions of the drawings, and a detailed description of the invention which is also referred to as the specification. Traditionally, patents are drafted by the patent attorneys who have extensive knowledge of both the law and the patent system, and it costs over $10K on average to draft a moderately complex patent (Quinn, 2015). Usually, the patent attorneys read the invention disclosure documents and interview the inventor(s) to understand the technical details of the invention, and then they draft the claims, drawings, and specification. Patent claims protect the boundaries of the invention, and hence, drafting claims requires the expertise of the patent attorneys. The drawings must follow the requirements of the patent office, and label every element of the drawing mentioned in the specification with a unique number. However, the bulk of patent text consists of specification, and the patent attorneys need to spend a significant amount of time and effort in drafting the specification to describe the invention in detail based on the claims and drawings. Figure 1 shows an example of a patent claim, relevant drawing, and the specification supporting the claim. In this work, we assume that the patent attorneys can provide their drafted claims and additional drawings as input to our system to automatically generate patent specification.

Transformer-based Large Language Models (LLMs) such as BERT (Devlin et al., 2019), T5 (Raffel et al., 2020), Gemini (Team et al., 2023), and GPT-3 (Brown et al., 2020) and its successor GPT-4 (Achiam et al., 2023) have shown impressive performances in the field of natural language generation. However, automating the generation of human-quality patent specification remains challenging for these LLMs, especially because patents are intricate legal documents that require each claim to be adequately supported in the specification, and the specification must describe the invention in sufficient details using the associated drawings. Hence, patents contain much more technical information than, e.g. general web text, making it difficult for the LLMs to capture all the relevant pieces of information pertaining to an invention to generate a coherent specification. Patent specification usually spans several pages, thus presenting another challenge for most of the LLMs which are limited by their token lengths, e.g., 512,

[1]https://www.uspto.gov/



Claim 7. The artificial intelligence-based apparatus of claim 2, wherein the processor is further configured to control an operation of the peripheral device according to the predicted average wake-up time or bed time.
Relevant specification: FIG. 4 is a configuration diagram of an artificial intelligence (AI)-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention. Referring to FIG. 4, the AI-based apparatus for providing wake-up and bed time information may include a communication unit 210a, a processor 260a, and a memory 230a. The processor 260a may control the operation of the peripheral device in correspondence to the predicted wake-up or bed time.
A brief description of the drawing: FIG. 4 is a configuration diagram of an artificial intelligence-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention.

Figure 1: An example of a patent drawing (left), claim, specification, and a brief description of the drawing (right).

1024, 2048, or 8192 tokens. Moreover, most pretrained LLMs are not trained on patent data, and thus cannot generate text in legal writing style.

In this paper, we present a novel method for generating patent specification to overcome the aforementioned limitations. We introduce two new tasks, *claim-to-specification* that simply generates specification for the given claim, and *claim+drawing-to-specification* that takes a claim and any associated drawing text as input to generate specification. To the best of our knowledge, our paper is the first work that generates patent specification by using claim and drawing text as inputs. We present new model-agnostic strategies to effectively construct training datasets with enriched context to help the model generate correct specification. We fine-tune two popular LLMs using our training datasets and show that our method can significantly outperform the pretrained and fine-tuned LLMs on several years of patent data. We also conduct an extensive user study to evaluate the *correctness* and *quality* of the generated specification and show that our proposed method can generate correct specification in 66% of the cases related to neural processor domain, and the quality of generated specification may sometimes be better than the actual, human-written, specification. We publicly release the first dataset for claim+drawing-to-specification task at `https://github.com/juriand/patentformer`.

2 Related Work

Most prior work related to patent text generation focused on generating specific sections of a patent, for example, Lee and Hsiang (2020a) generated patent claims by fine-tuning GPT-2, Lee (2020c) incorporated an additional BERT-based module on the basis of Lee and Hsiang (2020a) for personalizing the claims, Lee and Hsiang (2020b) presented a span-based approach and a generic framework to measure patent claim generation quantitatively, and Jiang et al. (2024) presented an approach to generate patent claims from detailed descriptions.

Lee (2020a) presented a text-to-text mapping approach for controlling patent text generation by using the structural metadata in patent documents, where the keywords in input indicate different generation tasks. Lee (2020b) presented approaches to control patent text generation by using semantic search. Lee (2023) pre-trained GPT-J model from scratch with patent corpus for autocompletion task and proposed a new metric called the Autocomplete Effectiveness (AE) ratio. Jieh-Sheng (2022) further improved upon the work of Lee (2023) by pre-training GPT-J-6B with patent text in both directions. Christofidellis et al. (2022) presented Patent Generative Transformer (PGT), a GPT-2 based model trained to facilitate part-of-patent generation-related tasks. Another line of related work focused only on summarizing patent text to generate short text, e.g., the title Souza et al. (2021), abstract Guoliang et al. (2023); Zhu et al. (2023), prior art Lee and Hsiang (2020c), or captions for patent figures Aubakirova et al. (2023).

Prior research on patent drafting either focused on generating a small section of text, for example, claims, or simply summarizing patent text to generate title or abstract. The closest related work is the study by Jiang et al. (2024) that generated claims from specification. To the best of our knowledge, our paper is the first work that generates patent specification from the claim and drawing text.

3 Methodology

Formally, let $\mathcal{P}$ represent a patent document containing a sequence of l claims, $\mathcal{C} = \{c_1, c_2, ..., c_l\}$, a sequence of m specification paragraphs, $\mathcal{S} = \{s_1, s_2, ..., s_m\}$, a set of t drawing images, $\mathcal{I} = \{i_1, i_2, ..., i_t\}$, and a set of t brief descriptions of the drawings, $\mathcal{B} = \{b_1, b_2, ..., b_t\}$, corresponding to each image in $\mathcal{I}$. For $\forall i_z \in I$, let n_z represent a set of k pairs of component names and their respective component numbers that appear in the drawing; $n_z = \{<i_{z_1}^{name}, i_{z_1}^{num}>, <i_{z_2}^{name}, i_{z_2}^{num}>, ..., <i_{z_k}^{name}, i_{z_k}^{num}>\}$, where $i_{z_j}^{name}$ is the name of j^{th} compo-

	$\mathcal{P}$	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{N}$
Mean	14.12K	1.49K	478.2	12.15K	274.0
Min	317	3	6	25	0
Max	4.56M	715.30K	276.29K	4.55M	24.91K
Std.	17.83K	1.2K	782.3	17.22K	335.0

Table 1: Number of tokens in various sections of patents that were granted by the USPTO from 2015 to 2023.

nent and $i_{z_j}^{num}$ is the number of j^{th} component in image i_z; $\mathcal{N}$ = $\{n_1, n_2, ..., n_t\}$ corresponding to all images in $\mathcal{I}$. Table 1 shows the average number of tokens in various sections, $\mathcal{P}$, $\mathcal{C}$, $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{N}$, of the 2M patents that were granted by the USPTO between 2015 and 2023.

3.1 Claim-to-Specification

First, we introduce the claim-to-specification task, $\mathcal{C}{\rightarrow}\mathcal{S}$. Capturing the claims and specification of an entire patent document into a single training example may not be possible for most LLMs due to their token length limits, since patents contain 14.12K tokens on average, as shown in Table 1. Moreover, learning from all the claims of an entire patent at once to produce the entire specification would be quite a challenging task for any model. So, we introduce an auxiliary task of mapping each claim feature to a paragraph in the specification, in order to fit most training samples within the 512 token length limit used by most LLMs.

Each claim, $c_x \in \mathcal{C}$, is either an independent claim or a dependent claim, and may describe multiple features of an invention, as described in detail in Appendix A.2. To make the task of matching claims to specification easier for the model, we first split each claim into one or more claim features and only keep the pairs <c_x, s_y> that have a cosine similarity of greater than or equal to 0.6 to ensure that only pairs with strong similarity are included in the training data. We provide more details in Appendix A.5. However, using cosine similarity can sometimes result in incorrect matching between claims and specification, so there is room for improvement in correctly matching c_x to s_y in the training data.

3.2 Claim+Drawing-to-Specification

Based on the task in Section 3.1, we then introduce an extended task called claim+drawing-to-specification, $\mathcal{T}{\rightarrow}\mathcal{S}$. Its goal is to generate output specification, $\mathcal{S}$, by using $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{N}$ as inputs, where the output specification must support the input claim features, $\mathcal{C}$, and correctly describe the drawings by using drawing descriptions, $\mathcal{B}$, and pairs of component names and numbers, $\mathcal{N}$, associated with each drawing.

We construct training samples containing the input and output pairs, <$\mathcal{T}$, $\mathcal{S}$>, where $\mathcal{T}$=<$\mathcal{C}, \mathcal{B}, \mathcal{N}$>. Similar to the claim-to-specification task, rather than learning from all the input text at once to produce the entire specification, we introduce an auxiliary task of mapping each claim feature to a paragraph in the specification and use only one drawing[2] associated with a paragraph.

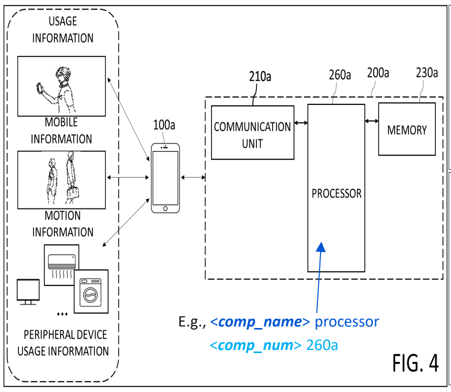
First, we match b_z to s_y by checking for common figure numbers. Then, we match s_y to c_x by using the methodology described in Section 3.1. Each $s_y \in \mathcal{S}$ may describe a figure or not. We only keep paragraphs that describe at least one figure in the patent by checking the presence of the words 'FIG.', 'Fig.', and 'Figure', as well as occurrences of any component names and numbers in each paragraph. Extracting n_z from the TIFF or PDF images, i_z, is not straightforward, so we instead extract the figure number, component names, and component numbers for each drawing from the specification, as described in Appendix A.4. Finally, we construct the quadruplets of samples, <c_x, b_z, n_z, s_y>, where <c_x, b_z, n_z> is the input to produce the corresponding output specification, s_y. We insert special tags into the input and output tokens to help the model with understanding different contexts.

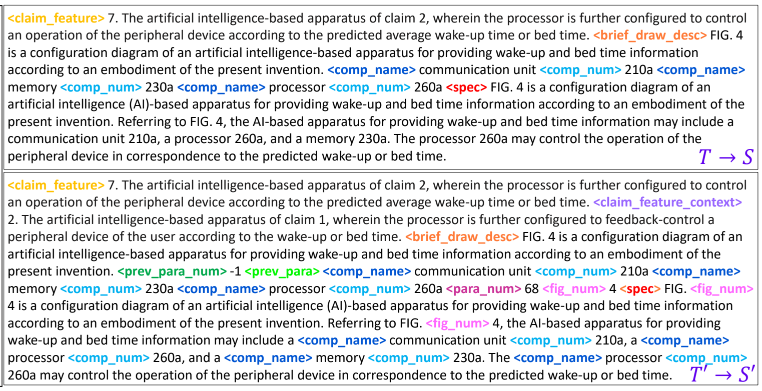
3.3 Patentformer

Now we introduce our method, Patentformer, that embeds rich context into the training data for generating specification. We design an enriched version of $\mathcal{T}$, represented as $\mathcal{T}'$=<$\mathcal{C}', \mathcal{B}', \mathcal{N}'$>, to generate $\mathcal{S}'$. Figure 2 shows a comparison between the training samples constructed for the claim+drawing-to-specification task, $\mathcal{T}{\rightarrow}\mathcal{S}$, and for Patentformer, $\mathcal{T}'{\rightarrow}\mathcal{S}'$, for the same example showed in Figure 1.

First, for each claim feature extracted from an independent claim, we provide as context the remaining claims features of that claim, and for each claim feature extracted from a dependent claim, we provide as context any remaining features of that claim as well as its parent claim as context. Second, for each figure number, component name, and component number, we embed special tags in the input as well as in the output specification to mark their presence in the training data. Third, we addi-

[2]Note that some paragraphs may describe more than one drawing. In this work, we assume that each paragraph describes only one drawing, and remove the lines from paragraph that refer to other figures, as described in Appendix A.3.




<claim_feature> 7. The artificial intelligence-based apparatus of claim 2, wherein the processor is further configured to control an operation of the peripheral device according to the predicted average wake-up time or bed time. <brief_draw_desc> FIG. 4 is a configuration diagram of an artificial intelligence-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention. <comp_name> communication unit <comp_num> 210a <comp_name> memory <comp_num> 230a <comp_name> processor <comp_num> 260a <spec> FIG. 4 is a configuration diagram of an artificial intelligence (AI)-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention. Referring to FIG. 4, the AI-based apparatus for providing wake-up and bed time information may include a communication unit 210a, a processor 260a, and a memory 230a. The processor 260a may control the operation of the peripheral device in correspondence to the predicted wake-up or bed time. $T \rightarrow S$

<claim_feature> 7. The artificial intelligence-based apparatus of claim 2, wherein the processor is further configured to control an operation of the peripheral device according to the predicted average wake-up time or bed time. <claim_feature_context> 2. The artificial intelligence-based apparatus of claim 1, wherein the processor is further configured to feedback-control a peripheral device of the user according to the wake-up or bed time. <brief_draw_desc> FIG. 4 is a configuration diagram of an artificial intelligence-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention. <prev_para_num> -1 <prev_para> <comp_name> communication unit <comp_num> 210a <comp_name> memory <comp_num> 230a <comp_name> processor <comp_num> 260a <para_num> 68 <fig_num> 4 <spec> FIG. <fig_num> 4 is a configuration diagram of an artificial intelligence (AI)-based apparatus for providing wake-up and bed time information according to an embodiment of the present invention. Referring to FIG. <fig_num> 4, the AI-based apparatus for providing wake-up and bed time information may include a <comp_name> communication unit <comp_num> 210a, a <comp_name> processor <comp_num> 260a, and a <comp_name> memory <comp_num> 230a. The <comp_name> processor <comp_num> 260a may control the operation of the peripheral device in correspondence to the predicted wake-up or bed time. $T' \rightarrow S'$

Figure 2: An example of a patent drawing (left), training data for claims+drawing-to-specification, $\mathcal{T}\rightarrow\mathcal{S}$, (top right), and enhanced training data, $\mathcal{T}'\rightarrow\mathcal{S}'$, for Patentformer (bottom right). Context tags are colored for readability.

tionally provide as context the previous paragraph, previous paragraph number, and current paragraph number to help the model with understanding various contexts to generate a coherent specification. We represent the enriched versions of $\mathcal{C}$, $\mathcal{N}$, and $\mathcal{S}$ as $\mathcal{C}'$, $\mathcal{N}'$, and $\mathcal{S}'$, respectively, and $\mathcal{B}'$=$\mathcal{B}$. Figure 2 shows the special tags associated with each context. As we will later show in Section 5.3, embedding rich context into the training data provides significant improvements to the model's performance.

4 Experimental Setup

In this section, we describe the dataset, models, and experimental settings to evaluate Patentformer.

Dataset. We construct the first dataset for generating specification from the claims and associated drawings. We worked with four patent experts and focused on patents in a specific CPC code, G06N[3], which includes patents from a diverse range of topics including artificial intelligence, neural networks, biological neurons, and artificial life, among many others. Figure 3 shows the t-SNE graph[4] for six main subcategories of G06N: (i) G06N 3/00: computing arrangements based on biological models, (ii) G06N 5/00: computing arrangements using knowledge-based models, (iii) G06N 7/00: computing arrangements based on specific mathematical models, (iv) G06N 10/00: quantum computing, i.e. information processing based on quantum-mechanical phenomena, (v) G06N 20/00: machine learning, and (vi) G06N 99/00: subject matter not provided for in other groups of this subclass.

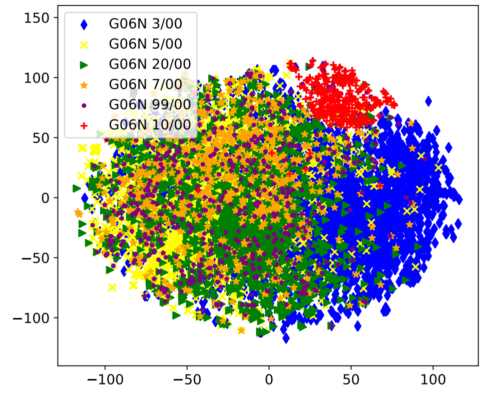


Figure 3: A t-SNE graph to visualize the relationships among patents in six subcategories of G06N CPC code.

Statistics (per patent)	Mean	Min	Max	Std.
# independent claims	8.33	0	110	6.49
# dependent claims	9.67	0	137	8.28
# claim features (with drawings)	18.00	1	153	12.03
# drawings	4.32	1	36	2.67

Table 2: Statistics of patent claims and drawings in the `Patent-2015-2023-G06N` dataset.

We used 13,725 patents in this category, representing about 0.69% of the total 2M patents that were granted by the USPTO between 2015 and 2023, to construct the `Patent-2015-2023-G06N` dataset consisting of 284,531 quadruplets[5] of <c_x, b_z, n_z, s_y>. Table 2 presents the average number of independent claims, dependent claims, claim features that are accompanied by a drawing, and drawings within patents in `Patent-2015-2023-G06N` dataset. In our experiments, two-thirds of the data

[3]`https://www.uspto.gov/web/patents/classification/cpc/html/defG06N.html#G06N`

[4]The t-SNE graph was computed by using a pre-trained Sentence Transformer, '`all-mpnet-base-v2`', from the Hugging Face to encode the titles and abstracts of 14,280 patents that were granted by the USPTO between 2015 and 2023.

[5]There were 3% paragraphs that describe a flow chart. Generating descriptions of flow charts is different from diagrams, because flow charts contain a series of steps and conditional statements. Thus, in this work, we do not focus on flow charts.

was used for training and one-third was reserved for evaluation. We truncated the text to fit within 512 and 2048 token limits for T5 and GPT-J models, respectively. On average, T5 (and GPT-J) models used 426 (and 479) input tokens and 199 (and 194) target tokens for training.

Models. For training Patentformer, we utilize two pre-trained models, a decoder-only based GPT-J model (Wang and Komatsuzaki, 2021) and an encoder-decoder based T5 (T5-11B) model (Raffel et al., 2020), and fine-tuned them on `Patent-2015-2023-G06N` dataset.

Baselines. Since this study presents the first work on generating specification from the claim and drawing text, there is no baseline from the literature for direct comparison, and hence, we follow prior art on using large pre-trained LLMs (T5 and GPT-J) in zero-shot setting without any further fine-tuning as baselines. We also designed a series of experiments to investigate the importance of each input text component, $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{N}$, as well as their context, on Patentformer's overall performance.

Performance Metrics. Since examining patents requires substantial expertise, we primarily rely on the human evaluations to judge the *correctness* and *quality* of the generated specification. To compare the models under various settings, we use the PPL (Perplexity) metric. We additionally report the performance of Patentformer using eleven popular metrics for natural language generation from the literature, including BLEU score and ROUGE scores (R-1, R-2, R-L, and R-Lsum), among others. We provide details on these metrics in Appendix B. We present the confidence interval (CI) for each model by using bootstrapping to select with replacement n samples from the test set (of size n) five times.

Training. We utilized NVIDIA A100 GPUs (80 GB per GPU) for model training. Each model was trained for 1 epoch with a batch size of 8 per device. Pat_T5* was trained for 2 epochs; justification for this choice is provided in Appendix C.

5 Results

In this section, we first compare the proposed Patentformer with chosen baselines using automatic evaluation metrics. Then, we present a user study to evaluate our method from the human's perspective. Finally, we perform an ablation study to show the effects of embedding rich context into training data on the performance of Patentformer.

5.1 Patentformer vs. Baselines

Table 3 presents the perplexity results for Patentformer and several baselines under various settings.

Pretrained vs. Fine-tuned LLMs. We first compare the pre-trained models, GPT-J (Pre) and T5 (Pre), with the same models after fine-tuning on patent text, Pat_GPT-J and Pat_T5, according to various tasks, $\mathcal{C}\rightarrow\mathcal{S}$, $\mathcal{T}\rightarrow\mathcal{S}$, and $\mathcal{T}'\rightarrow\mathcal{S}'$, as described in Section 3. Although the pre-trained models have learned to perform several NLP tasks by training on large text datasets, patent drafting is not included in these tasks. The special legal language in patents and the lack of knowledge of the downstream task make it difficult for the pretrained models to generate a reasonable specification. Therefore, as Table 3 shows, fine-tuning on patent data helps improve the performance of patent text generation.

Claim-to-specification. Specifically, fine-tuning the models on simple claim-to-specification task, $\mathcal{C}\rightarrow\mathcal{S}$, helps improve the performance, as shown in Table 3. However, the generated specification contains references to made-up figures, component names, and numbers, because this task lacks the drawing information.

Claim+drawing-to-specification. As we move to the extended task, $\mathcal{T}\rightarrow\mathcal{S}$, that utilizes both claim and drawing text, the quality of outputs highly improves. However, our proposed model, Patentformer, outperforms them by training on $\mathcal{T}'\rightarrow\mathcal{S}'$ task, which utilizes richer context for generation.

Post-processing generation strategy. We observed that the generated specification sometimes did not support the input claim, did not include the input component names and numbers, or incorrectly referred to other figures that were not presented to the model. To mitigate these issues, we implemented a simple post-processing step that

Model	PPL$^{\downarrow}$	95% CI	Training time
GPT-J (Pre)	12.353	12.363 $\pm$ 0.046	0
T5 (Pre)	4.072*10^6	4.025*10^6 $\pm$ 0.074*10^6	0
Pat_GPT-J ($\mathcal{C}\rightarrow\mathcal{S}$)	6.661	6.665 $\pm$ 0.017	27 hrs / 3 GPUs
Pat_T5 ($\mathcal{C}\rightarrow\mathcal{S}$)	6.003	6.000 $\pm$ 0.023	27 hrs / 4 GPUs
Pat_GPT-J ($\mathcal{T}\rightarrow\mathcal{S}$)	5.458	5.460 $\pm$ 0.011	27 hrs / 3 GPUs
Pat_T5 ($\mathcal{T}\rightarrow\mathcal{S}$)	4.649	4.645 $\pm$ 0.011	27 hrs / 4 GPUs
Pat_GPT-J ($\mathcal{T}'\rightarrow\mathcal{S}'$)	4.875	4.873 $\pm$ 0.007	27 hrs / 3 GPUs
Pat_T5 ($\mathcal{T}'\rightarrow\mathcal{S}'$)	3.790	3.789 $\pm$ 0.006	27 hrs / 4 GPUs
Pat_T5* ($\mathcal{T}'\rightarrow\mathcal{S}'$)	**3.771**	3.769 $\pm$ 0.005	54 hrs / 4 GPUs

Table 3: Comparison between the proposed model and several baselines. A lower PPL value is better. All PPL values fall within the 95% confidence interval.

Method	Pat_GPT-J_Greedy		Pat_T5*_Greedy		Pat_GPT-J_Top-kp		Pat_T5*_Top-kp		Pat_GPT-J_P		Pat_T5*_P	
	Score	95% CI	Score	95% CI	Score	95% CI	Score	95% CI	Score	95% CI	Score	95% CI
BERTScore	0.852	0.852 ±0.001	0.871	0.871 ±0.001	0.854	0.854 ±0.001	0.874	0.874 ±0.000	0.864	0.864 ±0.001	**0.878**	0.878 ±0.001
BLEU	0.164	0.164 ±0.002	0.239	0.238 ±0.003	0.146	0.146 ±0.002	0.234	0.234 ±0.002	0.179	0.178 ±0.001	**0.246**	0.245 ±0.003
ChrF	43.090	43.089 ±0.357	43.330	43.333 ±0.393	43.570	43.522 ±0.296	45.120	45.123 ±0.132	44.861	44.745 ±0.290	**46.533**	46.410 ±0.290
COMET	-0.589	-0.582 ±0.010	-0.344	-0.345 ±0.014	-0.403	-0.401 ±0.008	-0.190	-0.189 ±0.007	-0.220	-0.219 ±0.005	**-0.130**	-0.133 ±0.009
METEOR	0.350	0.350 ±0.003	0.370	0.369 ±0.003	0.352	0.351 ±0.003	0.380	0.379 ±0.002	0.372	0.372 ±0.003	**0.393**	0.392 ±0.003
NIST	4.213	4.156 ±0.032	5.872	5.757 ±0.066	4.091	4.034 ±0.022	6.097	5.977 ±0.021	4.856	4.765 ±0.012	**6.422**	6.278 ±0.044
R-1	0.409	0.410 ±0.002	0.486	0.486 ±0.004	0.423	0.424 ±0.001	0.499	0.499 ±0.001	0.460	0.461 ±0.002	**0.517**	0.516 ±0.002
R-2	0.218	0.219 ±0.003	0.295	0.294 ±0.003	0.205	0.205 ±0.002	0.284	0.284 ±0.001	0.239	0.239 ±0.002	**0.302**	0.301 ±0.003
R-L	0.296	0.297 ±0.002	**0.364**	0.363 ±0.003	0.272	0.272 ±0.002	0.346	0.346 ±0.001	0.299	0.298 ±0.002	0.360	0.359 ±0.002
R-Lsum	0.356	0.357 ±0.002	0.428	0.427 ±0.003	0.372	0.373 ±0.001	0.439	0.439 ±0.001	0.402	0.402 ±0.002	**0.456**	0.455 ±0.002
WER$^{\downarrow}$	1.266	1.260 ±0.006	1.001	0.997 ±0.006	1.287	1.283 ±0.008	0.966	0.966 ±0.004	1.170	1.168 ±0.005	**0.952**	0.951 ±0.007

Table 4: Comparison between greedy sampling, top-kp sampling, and post-processing strategy (_P) on 5000 test samples ($^{\downarrow}$ represents that a lower value is better). All scores, except NIST, fall within the 95% confidence interval.

ranks ten generated candidate specification paragraphs based on whether they describe the input claims using the input component names and numbers, and whether they contain reference(s) to other figures, as described in detail in Appendix D.

Since autoregressive generation with the post-processing step is computationally expensive[6], we show the results with post-processing in Table 4 using a small subset of 5000 samples from the `Patent-2015-2023-G06N` test data. We set the min/max limits as 100/512 for T5 and 50/256 for GPT-J; justification for this choice is provided in Appendix D. As Table 4 shows, our post-processing strategy outperforms both greedy and top-kp sampling on all metrics, except R-L. We present five random examples comparing the specification generated by Pat_T5* and Pat_GPT-J in Appendix F.

5.2 User Study

We worked with four patent experts who had extensive experience with drafting and reviewing patents in G06N category and asked them to judge a pair of two specification samples based on *correctness* and *quality*. We set a strict criteria for measuring the *correctness*: given a context claim, the claim feature must be supported in the specification *and* the specification must correctly refer to the component names and numbers of the associated drawing. *Quality* is the subjective opinion of the patent expert; we compared the quality only in cases where both the samples in a pair were marked as correct. In reality, the experts may have differing opinions about the correctness and quality of the generated specification, however, in our user study, each sample was evaluated by only one patent expert. Since reviewing specification requires extensive experience and knowledge of a particular technology, we selected a very small subset of patents from the `Patent-2015-2023-G06N` test dataset related to each patent expert's area of specialization. We used the post-processed versions of outputs from both Pat_T5* and Pat_GPT-J for the user study.

Study with random samples. We presented 100 pairs of random samples related to neural processor to one expert, and 40 pairs of random samples related to system-on-chip to another expert. While reviewing, the patent experts did not know which specification was model-generated and which one was true. We compared the correctness and quality of specification generated by Pat_T5* versus actual specification and Pat_GPT-J versus actual specification, and report the number of times each method wins/ties/loses (W/T/L) compared to the actual specification based on quality. Tables 5 and 6 present results of the experts' evaluation of random samples related to neural processor and system-on-chip technologies, respectively. As these results show, Pat_T5* was correct more often (33 out of 50 cases, 66%) than Pat_GPT-J (28 out of 50 cases, 56%) for neural processor related patents. For system-on-chip related patents, both Pat_T5* and Pat_GPT-J struggled to generate correct specification, however, Pat_T5* was correct more often (4 out of 20 cases, 20%) than Pat_GPT-J (2 out of 20 cases, 10%). This result also correlates with the better performance of Pat_T5* compared to Pat_GPT-J, as showed in Tables 3 and 4.

The patent experts marked many of the 'Actual' specification as incorrect due to incorrect matching among the elements of <c_x, b_z, n_z, s_y> quadruplet in the test data, and the low accuracy of 67% and 35% for the 'Actual' cases in Tables 5 and 6, respectively, indicates a huge room for improvement in aligning the claims, drawings, and specification paragraphs in the training set.

[6]It took 33 hours for Pat_T5* and 19 hours for Pat_GPT-J to generate 5000 samples using 1 Nvidia A100 GPU.

	Correctness		Quality
	# correct	# incorrect	W/T/L (vs. Actual)
Pat_T5*	33	17	15/6/9
Pat_GPT-J	28	22	14/5/7
Actual	67	33	N/A

Table 5: Correctness of Pat_T5*, GPT-J, and actual specification on 100 randomly selected patents related to neural processor. Quality of Pat_T5* and GPT-J in terms of Win/Tie/Loss (W/T/L) versus actual specification.

	Correctness		Quality
	# correct	# incorrect	W/T/L (vs. Actual)
Pat_T5*	4	16	0/1/2
Pat_GPT-J	2	18	0/0/1
Actual	14	26	N/A

Table 6: Correctness of Pat_T5*, GPT-J, and actual specification on 40 randomly selected patents related to system-on-chip. Quality of Pat_T5* and GPT-J in terms of Win/Tie/Loss (W/T/L) versus actual specification.

Study with two full patents. We next simulate the generation of specification for an entire patent with Pat_T5*. We asked two patent experts to review the actual versus model-generated specification for two randomly selected patents related to their areas of expertise. This time, we revealed to the experts which specification was generated by AI and which one was from the actual patent. Note that even though Patentformer was trained on both claim and drawing as input, in this realistic study, there were samples in which either the claim feature or the drawing was missing from the inputs. Even then, Pat_T5* generated correct specification in 53 out of 58 (91.38%) cases for one patent related to meta vision technology, but only 13 out of 81 (16.05%) cases for another patent related to memory technology. This result shows that Pat_T5* may not be directly applicable to all domains, and further fine-tuning may be required to achieve a desirable performance. We provide detailed results for the user study with two full patents in Appendix E.

5.3 Ablation Study

Next, we perform an ablation study to isolate the effects of adding various context to the training data on Patentformer's performance. Since Pat_T5 performed better than Pat_GPT-J (see Table 3), we conduct the ablation study with only Pat_T5.

Patent claims, drawings, and descriptions. We first remove the claims $\mathcal{C}'$, brief description of the drawings $\mathcal{B}'$, and components $\mathcal{N}'$ from the input, $\mathcal{T}'$, and evaluate the three models. As the top section of Table 7 shows, removing any one of

Model	PPL$^{\downarrow}$	95% CI
Pat_T5 ($\mathcal{T}' \to \mathcal{S}'$)	**3.790**	3.789 ± 0.006
Pat_T5 ($\mathcal{T}' - \mathcal{C}' \to \mathcal{S}'$)	4.818	4.815 ± 0.006
Pat_T5 ($\mathcal{T}' - \mathcal{B}' \to \mathcal{S}'$)	3.881	3.882 ± 0.009
Pat_T5 ($\mathcal{T}' - \mathcal{N}' \to \mathcal{S}'$)	5.488	5.478 ± 0.008
Pat_T5 ($\mathcal{T}'$ − Prev_Para $\to \mathcal{S}'$)	3.975	3.971 ± 0.006
Pat_T5 ($\mathcal{T}'$ − Prev_Para_Num $\to \mathcal{S}'$)	3.980	3.976 ± 0.007
Pat_T5 ($\mathcal{T}' \to \mathcal{S}'$ − Comp_Tags)	3.967	3.965 ± 0.007
Pat_T5 ($\mathcal{T}'$ − Para_Num $\to \mathcal{S}'$)	4.431	4.430 ± 0.005
Pat_T5 ($\mathcal{T}'$ − Fig_Num) $\to \mathcal{S}'$	3.849	3.851 ± 0.011
Pat_T5 ($\mathcal{T}'$ − Context_Claims) $\to \mathcal{S}'$	4.354	4.354 ± 0.005

Table 7: Results of the ablation study ($^{\downarrow}$ represents that a lower value is better). All PPL values, except for the setting Pat_T5 ($\mathcal{T}' - \mathcal{N}' \to \mathcal{S}'$), fall within the 95% confidence interval.

the inputs degrades model performance: removing components has the greatest impact, as the model generates incorrect component names and numbers that are inconsistent with the input drawings, and correcting such mistakes is infeasible for the users; removing claims also significantly degrades the model performance, as claims provide fundamental information for drafting the specification.

Rich context. We next study the effects of adding different contexts. As described in Section 3.3, we provided rich context to the model during training. As the bottom section of Table 7 shows, removing any context negatively affects the model's performance: removing paragraph numbers or context claims has the most effect; removing figure numbers results in a slight decrease in model performance, however, the model incorrectly refers to made-up figures. Similarly, removing the context of previous paragraph produces incoherent specification, and removing the component names and numbers injects incorrect components.

6 Conclusions

We proposed a novel method, Patentformer, to utilize diverse patent-related information, e.g., patent claims, drawing text, and brief descriptions of the drawings, for generating patent specification. We presented a model-agnostic approach to enrich the training dataset with richer context for the new *claim+drawing-to-specification* task. We evaluated our approach using both encoder-decoder and decoder-only LLMs and showed that our proposed method has the potential to generate correct specification in legal writing style. Human evaluation of the generated samples by four patent experts further affirmed the effectiveness and practical usefulness of our proposed method.

Limitations

Despite the shown capabilities of Patentformer, drafting a patent specification cannot be entirely automated, and the patent attorneys still need to thoroughly examine the generated specification to ensure both quality and correctness. For example, in the user study, the patent experts identified potential issues such as incorrect or inadequate descriptions of the claim features and inaccuracies in the drawing descriptions. The proposed method did not address generating specifications for special types of diagrams such as block diagrams or flow charts. Additionally, it was assumed that each specification paragraph would be associated with only *one* claim feature and *one* drawing, but in reality, a paragraph may be related to zero or more claim features and zero or more drawings. The model's performance can be improved by enhancing the alignment of claims, drawings, and specification paragraphs in the training set, adding special tags to handle different types of diagrams, and redesigning the training dataset to create quadruplets of training samples with zero or more drawings and zero or more claim features. In the future, we plan to leverage more advanced models that can handle larger number of tokens and larger contexts. Another line of future work is the exploration of multimodal models for patent text generation that can handle both drawing images and text inputs to generate specification.

Towards Deployment. In practice, the patent attorneys need to provide their drafted claims, drawings, descriptions of the drawings, and a correlation between the claim features and drawings. The system then needs to extract all the component names and numbers from each drawing file and ask the user to choose a set of component names and numbers from the drawing that are relevant to a given claim feature for generating specification.

Ethics Statement

We used publicly available patent data provided by the USPTO[7] to construct the `Patent-2015-2023-G06N` dataset. The user study reviews about quality are subjective views of the patent experts, and thus, the actual performance of Patentformer may be different than reported in this study. Patents are legal documents, and the USPTO[8] recommends the practitioners to take extra care to verify the technical accuracy of the documents and compliance with 35 U.S.C. 112 when using AI drafting tools (Holman, 2024).

Acknowledgements

We would like to express our sincere gratitude to the patent experts, Komal Magsi, Elliot Karlin, Kamil Bojanczyk, and Joseph Findley, for their guidance and expertise. Their valuable feedback while preparing the training datasets was instrumental in ensuring the accuracy and quality of the AI-generated patent specifications. Furthermore, their expert evaluations in the user studies were crucial in reviewing and comparing the AI-generated specifications against the actual specifications.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Dana Aubakirova, Kim Gerdes, and Lufei Liu. 2023. Patfig: Generating short and long captions for patent figures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2843–2849.

Laura Bergomi, Tommaso M. Buonocore, Paolo Antonazzo, Lorenzo Alberghi, Riccardo Bellazzi, Lorenzo Preda, Chandra Bortolotto, and Enea Parimbelli. 2024. Reshaping free-text radiology notes into structured reports with generative question answering transformers. *Artificial Intelligence in Medicine*, 154:102924.

Aanisha Bhattacharyya, Yaman K Singla, Balaji Krishnamurthy, Rajiv Ratn Shah, and Changyou Chen. 2023. A video is worth 4096 tokens: Verbalize videos to understand them in zero shot. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9822–9839, Singapore. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[7]`https://bulkdata.uspto.gov/`

[8]https://www.federalregister.gov/documents/2024/04/11/2024-07629/guidance-on-use-of-artificial-intelligence-based-tools-in-practice-before-the-united-states-patent

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2020. Cocon: A self-supervised approach for controlled text generation. *arXiv preprint arXiv:2006.03535*.

Dimitrios Christofidellis, Antonio Berrios Torres, Ashish Dave, Manuel Roveri, Kristin Schmidt, Sarath Swaminathan, Hans Vandierendonck, Dmitry Zubarev, and Matteo Manica. 2022. Pgt: a prompt based generative transformer for the patent domain. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11781–11791, Singapore. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Felix Faltings, Michel Galley, Kianté Brantley, Baolin Peng, Weixin Cai, Yizhe Zhang, Jianfeng Gao, and Bill Dolan. 2023. Interactive text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4450–4468, Singapore. Association for Computational Linguistics.

Yue Guo, Wei Qiu, Gondy Leroy, Sheng Wang, and Trevor Cohen. 2024. Retrieval augmentation of large language models for lay language generation. *Journal of Biomedical Informatics*, 149:104580.

Shi Guoliang, Zhou Shu, Wang Yunfeng, Shi Chunjiang, and Liu Liang. 2023. Generating patent text abstracts based on improved multi-head attention mechanism. *Data Analysis and Knowledge Discovery*, 7(6):61–72.

Christopher M Holman. 2024. The us patent and trademark office's response to recent developments in artificial intelligence. *Biotechnology Law Report*.

Lekang Jiang, Caiqi Zhang, Pascal A Scherz, and Stephan Goetz. 2024. Can large language models generate high-quality patent claims? *arXiv preprint arXiv:2406.19465*.

LEE Jieh-Sheng. 2022. The effectiveness of bidirectional generative patent language models. In *Legal Knowledge and Information Systems: JURIX 2022: The Thirty-fifth Annual Conference, Saarbrücken, Germany, 14-16 December 2022*, volume 362, page 194. IOS Press.

Mateusz Lango and Ondrej Dusek. 2023. Critic-driven decoding for mitigating hallucinations in data-to-text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2853–2862, Singapore. Association for Computational Linguistics.

Jieh-Sheng Lee. 2020a. Controlling patent text generation by structural metadata. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3241–3244.

Jieh-Sheng Lee. 2020b. Measuring and controlling text generation by semantic search. In *Companion Proceedings of the Web Conference 2020*, pages 269–273.

Jieh-Sheng Lee. 2020c. Patent transformer: A framework for personalized patent claim generation. In *CEUR Workshop Proceedings*, volume 2598. CEUR-WS.

Jieh-Sheng Lee. 2023. Evaluating generative patent language models. *World Patent Information*, 72:102173.

Jieh-Sheng Lee and Jieh Hsiang. 2020a. Patent claim generation by fine-tuning openai gpt-2. *World Patent Information*, 62:101983.

Jieh-Sheng Lee and Jieh Hsiang. 2020b. Patenttransformer-1.5: Measuring patent claim generation by span relevancy. In *New Frontiers in Artificial Intelligence*, pages 20–33, Cham. Springer International Publishing.

Jieh-Sheng Lee and Jieh Hsiang. 2020c. Prior art search and reranking for generated patent text. *arXiv preprint arXiv:2009.09132*.

Zi Liang, Pinghui Wang, Ruofei Zhang, Nuo Xu, Shuo Zhang, Lifeng Xing, Haitao Bai, and Ziyang Zhou. 2024. Merge: Fast private text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19884–19892.

Weiming Liao, Bo Chen, and Xiaobing Zhao. 2024. Zero-shot data-to-text generation via dual learning. In *Third International Conference on Electronic Information Engineering and Data Processing (EIEDP 2024)*, volume 13184, pages 778–782. SPIE.

Yupian Lin, Tong Ruan, Jingping Liu, and Haofen Wang. 2023. A survey on neural data-to-text generation. *IEEE Transactions on Knowledge and Data Engineering*.

Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15264–15281.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. 2024. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36.

Nicolo Micheletti, Samuel Belkadi, Lifeng Han, and Goran Nenadic. 2024. Exploration of masked and causal language modelling for text generation. *arXiv preprint arXiv:2405.12630*.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Yotam Perlitz, Ariel Gera, Michal Shmueli-Scheuer, Dafna Sheinwald, Noam Slonim, and Liat Ein-Dor. 2023. Active learning for natural language generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9862–9877, Singapore. Association for Computational Linguistics.

Gene Quinn. 2015. The cost of obtaining a patent in the us. Accessed: 2024-7-18.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Mohit Bansal, and Markus Dreyer. 2023. Generating summaries with controllable readability levels. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11669–11687, Singapore. Association for Computational Linguistics.

Furkan Şahinuç, Ilia Kuznetsov, Yufang Hou, and Iryna Gurevych. 2024. Systematic task exploration with LLMs: A study in citation text generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4832–4855, Bangkok, Thailand. Association for Computational Linguistics.

Cinthia M Souza, Magali RG Meireles, and Paulo EM Almeida. 2021. A comparative study of abstractive and extractive summarization techniques to label subgroups on patent dataset. *Scientometrics*, 126(1):135–156.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Sindhu Tipirneni, Ming Zhu, and Chandan K Reddy. 2024. Structcoder: Structure-aware transformer for code generation. *ACM Transactions on Knowledge Discovery from Data*, 18(3):1–20.

Dennis Ulmer, Chrysoula Zerva, and Andre Martins. 2024. Non-exchangeable conformal language generation with nearest neighbors. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1909–1929, St. Julian's, Malta. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Chunliu Wang, Huiyuan Lai, Malvina Nissim, and Johan Bos. 2023. Pre-trained language-meaning models for multilingual parsing and generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5586–5600, Toronto, Canada. Association for Computational Linguistics.

Manjeet Yadav, Nilesh Kumar Sahu, Mudita Chaturvedi, Snehil Gupta, and Haroon R Lone. 2024. Fine-tuning large language models for automated diagnostic screening summaries. *arXiv preprint arXiv:2403.20145*.

Haibin Yu, Yuxuan Hu, Yao Qian, Ma Jin, Linquan Liu, Shujie Liu, Yu Shi, Yanmin Qian, Edward Lin, and Michael Zeng. 2023. Code-switching text generation and injection in mandarin-english asr. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Feng Zhao, Hongzhi Zou, and Cheng Yan. 2023a. Structure-aware knowledge graph-to-text generation with planning selection and similarity distinction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8693–8703.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023b. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.

Changsheng Zhu, Xin Zheng, and Wenfang Feng. 2023. An automatic generation method of patent specification abstract based on" extraction-abstraction" model. In *2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA)*, pages 196–200. IEEE.

A Patent Documents

A patent document usually consists of the title, abstract, field of the invention, background, summary of the invention, independent claims, dependent claims, drawings, brief descriptions of the drawings, and a detailed description of the invention which is also referred to as the specification. A patent document may additionally contain drawings to help describe the invention, and the drawings must follow the standards of the patent office and label every element of the drawing that is mentioned in the specification with a unique number for identification. We next describe each section of a patent in detail.

A.1 The Title and the Abstract

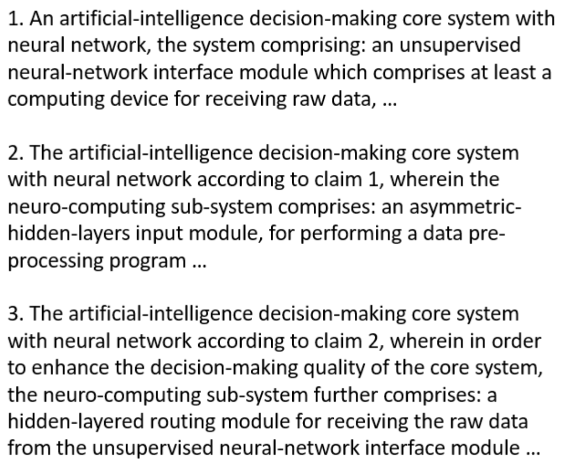
The title of a patent is a generic summary of the invention based on the claims of the invention. The abstract is a summary of the invention based on the description, claims, and any drawings, and it clearly explains the technical problem, the proposed solution of the problem through the invention, and the use(s) of the invention.

A.2 Claims

Patent claims are the most essential part of a patent application, because claims protect the boundaries of the invention. A claim can be either an independent claim or a dependent claim, as described next. A claim that stands alone and describes all the possible limitations necessary to define an invention is called an independent claim. A dependent claim refers to a previous claim and must add a further feature or limitation to the previous claim. Examples of an independent claim and a dependent claim are provided in Figure 4.

Claim subtrees. Claims may comprise of multiple sentences in a hierarchical structure, where each sentence is a claim, which might be dependent on other claims. For each claim, we construct the claim subtrees consisting of two nodes, the node itself and its ancestor node. For example, in Figure 5, claim 1 is the ancestor of claim 2, claim 2 is the ancestor of claim 3, and so on.

Claim feature. Each claim may describe one or more features of the invention, and the claim features are usually separated by a semicolon in the claims. We observed that each paragraph in the specification may not fully cover the entire claim, so we further extracted the claim features



1. An artificial-intelligence decision-making core system with neural network, the system comprising: an unsupervised neural-network interface module which comprises at least a computing device for receiving raw data, ...

2. The artificial-intelligence decision-making core system with neural network according to claim 1, wherein the neuro-computing sub-system comprises: an asymmetric-hidden-layers input module, for performing a data pre-processing program ...

3. The artificial-intelligence decision-making core system with neural network according to claim 2, wherein in order to enhance the decision-making quality of the core system, the neuro-computing sub-system further comprises: a hidden-layered routing module for receiving the raw data from the unsupervised neural-network interface module ...

Figure 4: Claim 1 is an independent claim. Claim 2 is dependent on claim 1, and claim 3 is dependent on claim 2.

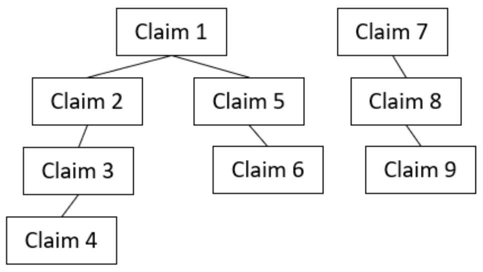


Figure 5: An example of claim subtrees.

from each claim. For simplicity, we separated the claims by using a semicolon to extract one or more claim features from a claim. We then matched each claim feature to a paragraph in specification, which resulted in a better overall matching between the claims and specification. For Patentformer, for each claim feature that is extracted from an independent claim, we provide as context any remaining claims features of that claim, and for each claim feature that is extracted from a dependent claim, we provide as context any remaining claim features of that claim as well as its parent claim as context.

A.3 Specification

Patent specification describes the invention in detail based on the claims and any associated drawings. Patent specification usually starts by describing the field of the invention, the background, and a summary of the invention. Then, it focuses on describing the various aspects of the invention based on the claims and associated drawings. In this work, we only focused on generating parts of specification that describe the claims as well as associated drawings, as explained in the claim+drawing-to-specification task introduced in this study. USPTO patent data[9] consists of specification paragraphs, where some paragraphs are too short and may not fully describe a given claim feature. So, we concatenated each paragraph that was shorter than 50 tokens with its subsequent paragraph. We truncated longer paragraphs to fit within the 512 and 2048 token length limits of T5 and GPT-J models, respectively.

Drawing. In this study, we extracted the figure number of each drawing by checking for the presence of 'FIG.', 'Fig.', and 'Figure', as well as occurrences of any component names and numbers within specification paragraphs. We noticed that a subsequent paragraph may continue describing the same figure without explicitly mentioning any figure number or component numbers. Therefore, we kept up to two paragraphs after each paragraph that mentioned a figure, and mapped the same figure number to the following two paragraphs. We noticed that sometimes a paragraph may describe more than one figure. In this work, we focused on generating specification for only one claim feature and one drawing, so we explicitly removed the sentences from each paragraph that mentioned any other figure numbers.

A.4 Component Names and Numbers

Rather than extracting the component names and their respective numbers from the image files provided by the USPTO in TIFF and PDF formats, we simulated the extraction of component names and numbers for each drawing by using the specification paragraphs, as described next. The USPTO patent data contains specification paragraphs with special tags for the component numbers, however, we need to extract the component names from the specification text. So, we ran the longest common substring algorithm to find the component name for the sequences of text that end with the same component number.

A.5 Claim-to-Specification

In order to compute similarity between a given claim feature and specification paragraph, we used a pre-trained Sentence Transformer model, '`multi-qa-mpnet-base-dot-v1`', from the Hugging Face to compute the embeddings for both claim feature and specification. We then computed cosine similarity between the normalized embeddings of both the claim feature and specification, and discarded any claim-specification pair that had a cosine similarity of less than 0.6, in order to control the quality of training data.

[9]https://bulkdata.uspto.gov/

B Performance Metrics

Since examining patent specification requires the expertise of patent attorneys, we primarily rely and focus on human evaluations to test the performance of Patentformer. Since this is the first work that generates patent specification from claim and drawing text, we did not know which metrics for automated evaluation would be the best to evaluate the quality of generated patent specification against true specification, so we surveyed 28 recently published papers (Vaswani et al., 2017; Radford et al., 2019; Brown et al., 2020; Lee, 2020a; Zhao et al., 2023b; Deng and Raffel, 2023; Faltings et al., 2023; Zhao et al., 2023a; Lango and Dusek, 2023; Ribeiro et al., 2023; Liu et al., 2023; Perlitz et al., 2023; Bhattacharyya et al., 2023; Lovelace et al., 2024; Guo et al., 2024; Liang et al., 2024; Tipirneni et al., 2024; Şahinuç et al., 2024; Munkhdalai et al., 2024; Liao et al., 2024; Micheletti et al., 2024; Bergomi et al., 2024; Yu et al., 2023; Lin et al., 2023; Yadav et al., 2024; Wang et al., 2023; Ulmer et al., 2024; Chan et al., 2020) related to natural language generation, and used the twelve most popular metrics from these papers in our study, as described next.

We evaluated the performance of Patentformer across the following twelve popular metrics for natural language generation. Perplexity measures the probability of a reference sentence to be produced by the model. BLEU (BiLingual Evaluation Understudy) score measures the similarity between a reference text and the model generated text. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures how much of the important content from reference text matches with the model-generated text. We used four variants of ROUGE metric, namely ROUGE-1 (unigram based scoring), ROUGE-2 (bigram based scoring), ROUGE-L (longest common subsequence based scoring), and ROUGE-LSum (average of ROUGE-L score for each sentence). Word Error Rate (WER) counts the minimum number of edits needed to change the generated text to match the reference text. NIST (National Institute of Standards and Technology) is derived from BLEU score but it additionally considers how informative a particular n-gram is. ME-

Model	Num Epochs	PPL$^{\downarrow}$	Training times
Pat_GPT-J	1	**4.875**	27 hours / 3 GPUs
Pat_GPT-J	2	5.662	54 hours / 3 GPUs
Pat_GPT-J	3	6.646	81 hours / 3 GPUs
Pat_GPT-J	4	8.002	108 hours / 3 GPUs
Pat_T5	1	3.790	27 hours / 4 GPUs
Pat_T5	2	**3.771**	54 hours / 4 GPUs
Pat_T5	3	4.041	81 hours / 4 GPUs
Pat_T5	4	3.893	108 hours / 4 GPUs

Table 8: Ablation study with various epochs. ($^{\downarrow}$ represents that a lower value is better).

TEOR (Metric for Evaluation of Translation with Explicit ORdering) measures the quality of generated text based on the alignment between the generated text and a reference text, by computing the harmonic mean of unigram precision and recall, with recall weighted higher than precision. ChrF (CHaRacter-level F-score) calculates the similarity between the generated text and a reference text by using character n-grams, not word n-grams. BERTScore (Zhang et al., 2019) measures the semantic similarity between the generated text and a reference text by using sentence representations from BERT model. COMET (Crosslingual Optimized Metric for Evaluation of Translation) (Rei et al., 2020) is similar to BERTScore, but is trained to predict quality scores for translations.

C Results with More Epochs

In this section, we present the results for training both Pat_T5 and Pat_GPT-J on up to four epochs. As Table 8 shows, training Pat_GPT-J with more than 1 epoch degrades the model performance. The performance of Pat_T5 improved with 2 epochs, so we chose Pat_T5* with 2 epochs in the main paper.

D Post-processing Strategy

Generally, GPT-J produced longer outputs (1554 tokens on average) and T5 produced shorter outputs (181 tokens on average). So, for a fairer comparison during generation, we set the min/max limits as 100/512 tokens for T5 and 50/256 tokens for GPT-J. This resulted in 174 tokens on average for T5 and 206 tokens on average for GPT-J during generation, which are closer to the average token lengths of specification paragraphs in the training data (199 tokens for T5 and 194 tokens for GPT-J).

We observed that the generated specification sometimes did not support the input claim, did not include the input component names and numbers, or incorrectly referred to other figures that were not presented to the model. To mitigate these issues, we implemented a simple post-processing step that ranks ten generated candidate specification paragraphs based on a scoring function, $\mathcal{F} = argmax_i(f_i^1 + f_i^2 + f_i^3)$, where:

$$f_i^1 = cosim(c_x, \hat{s}_i)$$

$$f_i^2 = \begin{cases} 1, \text{ if no input components} \\ \frac{|\widehat{\mathcal{N}} \cap \mathcal{N}|}{|\mathcal{N}|}, \text{ otherwise} \end{cases} \tag{1}$$

$$f_i^3 = \begin{cases} 1, \text{ if no reference to other figures} \\ 0, \text{ otherwise} \end{cases}$$

where, $\widehat{\mathcal{N}}$ is the set of component names and numbers in the generated specification, $\hat{s}_i$. And, $cosim(c_x, \hat{s}_i)$ calculates the cosine similarity between c_x and $\hat{s}_i$, by using their embeddings from the pre-trained Sentence Transformer model, '`multi-qa-mpnet-base-dot-v1`'.

E User Study

In the main paper, we presented the user study results with 100 random samples related to neural processor domain, and 40 random samples related to system-on-chip domain. In this section, we present the detailed results for study with two patents related to meta vision and memory technologies in Tables 9 and 10, respectively.

In the study with two full patents, there were cases where either a claim feature or drawing was missing from the inputs. Specifically, the meta vision related patent containing a total of 58 samples had 11 samples without a matching figure and 35 samples without a matching claim feature; the memory technology related patent containing a total of 81 samples had 12 samples without a matching figure and 17 samples without a matching claim feature. Even then, Pat_T5* generated correct specification in 53 out of 58 (91.38%) cases for the meta vision related patent, but only 13 out of 81 (16.05%) cases for the memory technology related patent.

F Actual versus Patentformer-generated Specification

In this section, we present five random examples of patent specification generated by Pat_GPT-J and Patentformer, Pat_T5*, and compare them with the actual specification in Tables 11, 12, 13, 14 and 15. Note that even though we showed the

	Correctness		Quality
	# correct	# incorrect	W/T/L (vs. Actual)
Pat_T5*	53	5	23/11/11
Actual	49	9	N/A

Table 9: Correctness and quality of one randomly selected patent from 'G06N' category related to meta vision technology. Correctness of Pat_T5* versus and Actual data, and quality, in terms of W/T/L (Win/Tie/Loss) counts, of Pat_T5* compared to the actual specification.

	Correctness		Quality
	# correct	# incorrect	W/T/L (vs. Actual)
Pat_T5*	13	68	0/0/6
Actual	18	63	N/A

Table 10: Correctness and quality of one randomly selected patent from 'G06N' category related to memory technology. Correctness of Pat_T5* versus and Actual data, and quality, in terms of W/T/L (Win/Tie/Loss) counts, of Pat_T5* compared to the actual specification.

associated drawing image in these tables to the patent experts during user study, we did not utilize the image modality. Using multi-modal models for incorporating both patent image and text is not orthogonal to work, and may improve upon our work, however, in this work, we focused on using only the text from the drawings.

G Towards Deployment

We presented a novel method to generate specification from the input claims and drawings. The goal of the system is to assist the users to enter their drafted claims, drawings, and brief descriptions of the drawings. The system then helps the user to map the claim features to zero or more drawings, after which they can begin drafting the specification using Patentformer, as described next. At inference time, the system utilizes the input claim features, drawing text, and mappings between the claim features and drawings to produce the output specification. An application of this system is to assist the patent attorneys with generating specific paragraphs based on the provided inputs. For instance, when the user starts typing a figure number, e.g., 'FIG. 2', the system can display the relevant claim features ($\mathcal{C}$) and components related to that figure. We assume that the system can extract the component names and numberings ($\mathcal{N}$) from the input figure by employing OCR or parsing the text from powerpoint/Visio/DWG/etc. files containing the figures. Since the system is already aware of the brief descriptions of drawings, $\mathcal{B}$, as previously provided by the user, it can leverage $\mathcal{B}$ as additional input to generate specification for the corresponding figure and claim feature.

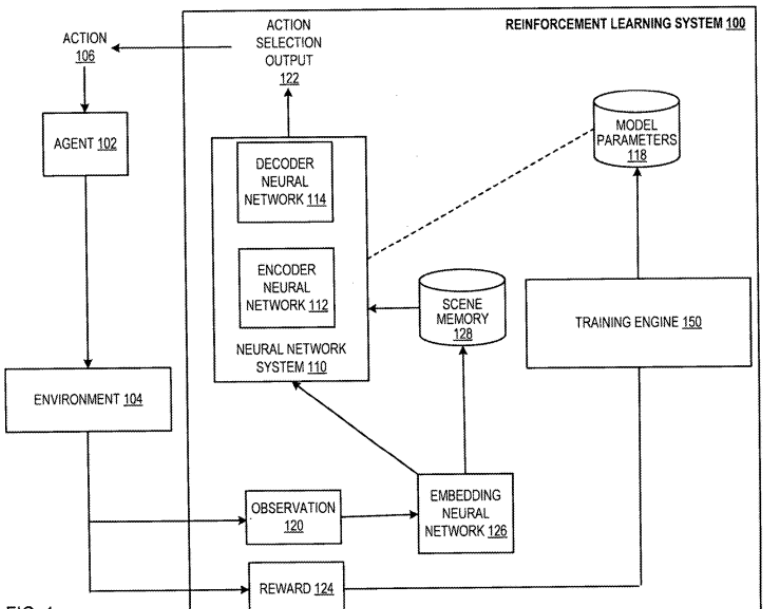
Input Figure	
 ACTION 106 ACTION SELECTION OUTPUT 122 REINFORCEMENT LEARNING SYSTEM 100 AGENT 102 MODEL PARAMETERS 118 DECODER NEURAL NETWORK 114 ENCODER NEURAL NETWORK 112 SCENE MEMORY 128 TRAINING ENGINE 150 NEURAL NETWORK SYSTEM 110 ENVIRONMENT 104 EMBEDDING NEURAL NETWORK 126 OBSERVATION 120 REWARD 124 FIG. 1	
Input Text:	**<claim_feature>** 7. The method of claim 6, wherein generating updated scene memory data comprises: clustering (i) the embeddings of the prior observations and (ii) a compressed representation of the embeddings of the prior observations into a plurality of clusters. **<claim_feature_context>** 6. The method of claim 3, wherein the compressed representation comprises a plurality of center embeddings, wherein each center embedding is a center of a respective cluster of the embeddings in the scene memory data. and generating a new compressed representation that includes a respective center embedding for each of the plurality of clusters. **<brief_draw_desc>** FIG. 1 shows an example control system. **<prev_para_num>** 46 **<prev_para>** In particular, instead of including the encoder attention block described above, the encoder neural network 112 instead includes two attention blocks and the scene memory data includes (i) embeddings and (ii) a compressed representation of the embeddings. The compressed representation generally includes many fewer embeddings than the total number of prior observations. **<para_num>** 47 **<fig_num>** 1 **<spec>**
Actual	In particular, the compressed representation includes a plurality, i.e., a fixed number, of center embeddings. Each center embedding is the center of a respective cluster of embeddings of the prior observations. The system can cluster the embeddings in the scene memory data and identify the centers of those clusters using any of a variety of existing clustering techniques. For example, the system can apply iterative farther point sampling on the embeddings to determine the centers.
Pat_T5*	The scene memory data includes a plurality of clusters, where each cluster corresponds to a respective set of prior observations. The system generates updated scene memory data by: (i) clustering (i) the embeddings of the prior observations and (ii) a compressed representation of the embeddings of the prior observations into a plurality of clusters, and (iii) generating a new compressed representation that includes a respective center embedding for each of the plurality of clusters.
Pat_GPT-J	The system generates (208) a compressed representation of the current observation (the embeddings of the current observation). The system generates (210) updated scene memory data based on the current observation. Updating the scene memory data can include generating a plurality of cluster center representations of the scene memory data and identifying the clusters in the scene memory data using the plurality of cluster center representations. Generating the scene memory data can include, for each cluster in the scene memory data, setting the value of the cluster to a respective cluster embedding and generating an updated center embedding for the cluster by computing a weighted sum of the cluster embeddings in the scene memory data. The system can generate an initial scene memory data, e.g., for the first prior observation, by pre-training the encoder neural network. In some implementations, the initial scene memory data includes the cluster centers that are provided by the encoder neural network and the cluster centers for the training observations. Generating the updated scene memory data can include, for each cluster in the scene memory data, setting the value of the cluster to the updated center embedding for the cluster. Updating the cluster centers can include computing a weighted sum of the cluster embeddings.

Table 11: An example of patent specification generated by Patentformer, Pat_T5*, and Pat_GPT-J compared to the actual specification. The input text was embedded with rich context to generate the output specification.

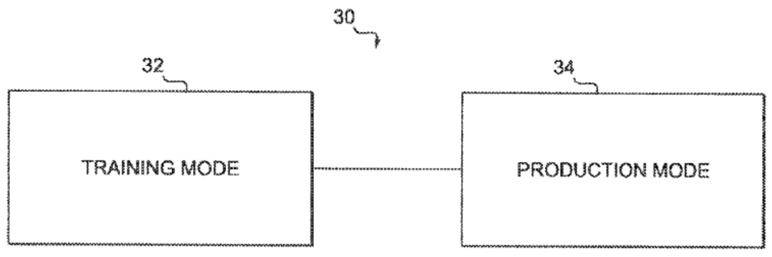
Input Figure	
 FIG. 1	
Input Text:	**<claim_feature>** 34. A method as in claim 33, further comprising the training mode using the computer to store quantities associated with a new approximate blood flow through the perturbed at least one heart valve, inflow tract or outflow tract. **<claim_feature_context>** 33. A method as in claim 32, further comprising the training mode using the computer to calculate a new approximate blood flow through the at least one heart valve, inflow tract or outflow tract with the perturbed at least one known anatomical characteristic or known physiological characteristic. **<brief_draw_desc>** FIG. 1 is a block diagram of a machine learning system, according to one embodiment; **<prev_para_num>** -1 **<prev_para>** **<comp_name>** system **<comp_num>** 30 **<comp_name>** training mode **<comp_num>** 32 **<comp_name>** production mode **<comp_num>** 34 **<para_num>** 2 **<fig_num>** 1 **<spec>**
Actual	Referring to FIG. 1 and according to one embodiment, a machine learning system 30 may include two modes: a training mode 32 and a production mode 34. The two modes 32, 34 may be embodied in a computer system and/or a computer readable medium. The system 30 may execute the two modes in series, where the training mode 32 is executed first, and the production mode 34 is executed second. The training mode 32 may be configured to develop analytical capabilities in a computer system that enable the computer system to predict unknown anatomic and/or physiologic characteristics of one or more heart valves and/or the surrounding inflow/outflow tracts. These predictive capabilities may be developed by the analysis and/or evaluation of known anatomic and/or physiologic characteristics of one or more heart valves and/or the surrounding inflow/outflow tracts. Using a collection of known anatomic and/or physiologic characteristics, a computer may be "trained" to predict various unknown anatomic and/or physiologic characteristics. The abstract mapping that transforms a set of known characteristics into one or more predictions of unknown characteristics may be referred to as the "transformation function." In some embodiments, the training mode 32 may be configured to construct the transformation function.
Pat_T5*	FIG. 1 shows an example machine learning system 30, according to one embodiment. The system 30 includes a training mode 32 and a production mode 34, which may be switched between each other at any time. The training mode 32 is used to train the system 30, for example by changing at least one of the anatomical characteristics or physiological characteristics of at least one of the valves. The training mode 32 is used to store quantities associated with the calculated or estimated blood flow through the heart valve based on the perturbed anatomical characteristic or physiological characteristic. Generally speaking, during the training mode, the system trains the model to estimate blood flow through the heart valve based on at least one of the anatomical characteristic or physiological characteristic that is manipulated, in the training mode. The training mode 32 is used to store quantities associated with the calculated or estimated blood flow through the heart valve based on the perturbed anatomical characteristic or physiological characteristic. The training mode 32 is used to store quantities associated with the calculated or estimated blood flow through the heart valve based on at least one of the anatomical characteristic or physiological characteristic that is manipulated, in the training mode.
Pat_GPT-J	In system 30, depicted in FIG. 1, training mode 32 uses a computer to calculate a new approximate blood flow through a patient's heart valve(s), inflow tract(s) and/or outflow tract(s) using at least one known anatomical characteristic or known physiological characteristic of the patient. The computer can be part of a server, desktop computer or other computer system and the system 30 can be in the form of a stand-alone unit or part of a larger computer system. In production mode 34, the computer can be used to calculate a new approximate blood flow through the patient's at least one heart valve, inflow tract(s) or outflow tract(s), using the new calculated approximate blood flow through the at least one heart valve, inflow tract(s) or outflow tract(s). In production mode 34, the computer system can be part of a network of computers, such as a healthcare cloud and at least some of the steps described below can be performed in a central location or in each individual patient's healthcare system.P1:Q3

Table 12: An example of patent specification generated by Patentformer, Pat_T5*, and Pat_GPT-J compared to the actual specification. The input text was embedded with rich context to generate the output specification.

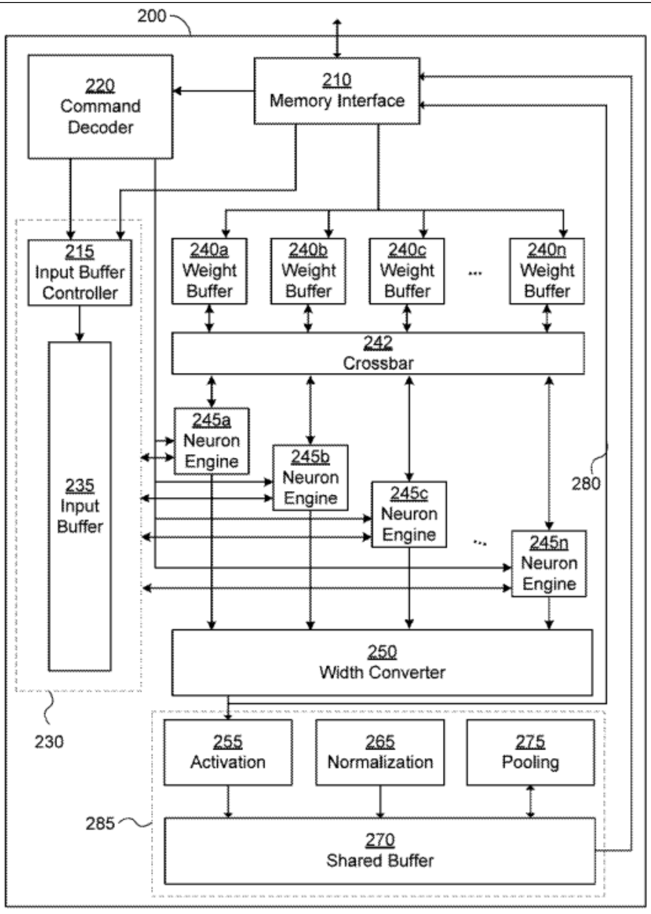
Input Figure	
 200 220 Command Decoder 210 Memory Interface 215 Input Buffer Controller 240a Weight Buffer 240b Weight Buffer 240c Weight Buffer 240n Weight Buffer 242 Crossbar 235 Input Buffer 245a Neuron Engine 245b Neuron Engine 245c Neuron Engine 245n Neuron Engine 280 250 Width Converter 230 255 Activation 265 Normalization 275 Pooling 285 270 Shared Buffer FIGURE 2	
Input Text:	**<claim_feature>** 2. Hardware as claimed in claim 1, wherein the number of weights in a filter is the same as the number of data values in a data window. **<claim_feature_context>** 1. Hardware for implementing a Deep Neural Network (DNN) having a convolution layer, the hardware comprising: an input buffer configured to provide data windows to a plurality of convolution engines, each data window comprising a single input plane; and each of the plurality of convolution engines being operable to perform a convolution operation by applying a filter to a data window, each filter comprising a set of weights for multiplication with respective data values of a data window, and each of the plurality of convolution engines comprising: multiplication logic operable to multiply a weight of the filter with a respective data value of the data window provided by the input buffer; and accumulation logic configured to accumulate the results of a plurality of multiplications performed by the multiplication logic so as to form an output for a respective convolution operation. **<brief_draw_desc>** FIG. 2 is a schematic diagram of hardware for implementing a Deep Neural Network; **<prev_para_num>** 107 **<prev_para>** However, when there is extremely high sparsity (e.g. almost all of the data or weights are zero) the overhead of fetching the next set of data/weights may not be hidden by the non-zero filter operations performed by the neuron engines. An improved approach to performing depthwise separable convolutions using the neuron engine approach set out herein will now be described. **<comp_name>** hardware **<comp_num>** 200 **<para_num>** 108 **<fig_num>** 2 **<spec>**
Actual	The hardware 200 may be configured in a single plane filter mode in which filters are performed in a window comprising a single plane. In a typical configuration of a convolution filter, each window in the data contains all input planes, and the filter contains the same number of weights as the number of elements in the window of depth P planes. In single plane filter mode each window is instead defined as containing only a single input plane, and (if necessary) separate windows are defined for each input plane. In this case the number of weights in a filter is the same number as the number of elements in the single plane window. This significantly reduces the number of weights that need to be stored and transferred in order to perform a filter on a given window.
Pat_T5*	In some embodiments of the hardware 200, the number of weights in a filter is the same as the number of data values in a data window, e.g. the filters are 11 filters. The number of neuron engines required to perform the convolution may therefore be determined by the number of weights in a filter. For example, a filter may comprise a filter for each of the 12 data planes assuming that the depth of the neural network is eight.
Pat_GPT-J	In one example the number of weights in a filter is the same as the number of data values in a data window. Thus in some examples each filter (e.g. convolution kernel) is the same size as the data window. As each filter is the same size then so is the amount of data (e.g. input planes) needed to be provided to the convolution engines at one time. For example, for a hardware 200 as described herein having four sets of neuron engines then each convolution kernel (e.g. filter) needs to be the same size as the input planes to be convolved (i.e. multiplied with). If the number of filters is F then the number of convolution engines that need to be fed with the same number of input planes will be F. Thus, it will be appreciated that it will no longer be necessary to fetch a plurality of input planes of data when the data to be convolved is highly sparse.

Table 13: An example of patent specification generated by Patentformer, Pat_T5*, and Pat_GPT-J compared to the actual specification. The input text was embedded with rich context to generate the output specification.

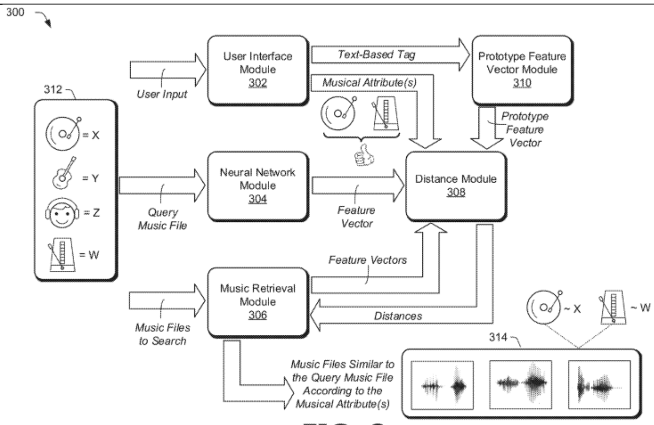
Input Figure	
 300 User Interface Module 302 Text-Based Tag Prototype Feature Vector Module 310 Musical Attribute(s) User Input 312 = X = Y = Z = W Prototype Feature Vector Neural Network Module 304 Distance Module 308 Query Music File Feature Vector Feature Vectors Music Retrieval Module 306 Distances Music Files to Search ~ X ~ W 314 Music Files Similar to the Query Music File According to the Musical Attribute(s) FIG. 3	
Input Text:	**<claim_feature>** 3. The method as described in claim 1, further comprising providing audio content of the similar audio files for playback by the computing device. **<claim_feature_context>** 1. In a digital medium environment for searching for music, a method implemented by a computing device, the method comprising: receiving a music search request that includes a music file including music content; receiving a selected musical attribute from a plurality of musical attributes; generating musical features of the music content, a respective one or more of the musical features corresponding to a respective one of the plurality of musical attributes; comparing the respective one or more musical features corresponding to the selected musical attribute to audio features of a plurality of audio files; and determining similar audio files to the music file based on the comparing of the respective one or more musical features to the audio features of the plurality of audio files. **<brief_draw_desc>** FIG. 3 illustrates an example system usable for searching for music in accordance with one or more aspects of the disclosure. **<prev_para_num>** 61 **<prev_para>** – **<comp_name>** computing device **<comp_num>** 102 **<comp_name>** music search system **<comp_num>** 104 **<comp_name>** query music file **<comp_num>** 112 **<comp_name>** results panel **<comp_num>** 130 **<comp_name>** first music file **<comp_num>** 132 **<comp_name>** second music file **<comp_num>** 134 **<comp_name>** third music file **<comp_num>** 136 **<para_num>** 62 **<fig_num>** 3 **<spec>**
Actual	A user can select any of the first music file 132, the second music file 134, or the third music file 136, and play the music content of the selected file. For instance, a user can click on the displayed representation of one of the music files in the results panel 130 and cause the computing device 102 to play the music content of the selected music file through a loudspeaker of the computing device 102. In one example, a user can select one of the first music file 132, the second music file 134, or the third music file 136 and designate the selection as a new query music file (e.g., to replace the query music file 112). The user can then conduct another search for music files that have music content perceptually similar to the music content of the new query music file. Hence, the user may continue to refine the search in an iterative way, by designating a music file returned by the music search system 104 as a new query music file for a next iteration in the search.
Pat_T5*	For instance, the computing device 102 may be configured to display an audio content of the first music file 132, the second music file 134, and the third music file 136, as indicated in the results panel 130, for playback by a user of the computing device 102. As an example, the computing device 102 may receive a user input request to play the query music file 112. Based on the user input request, the music search system 104 may provide audio content of the first music file 132, the second music file 134, and the third music file 136 to the computing device 102 for playback by a user of the computing device 102. For instance, the user can select one of the first music file 132, the second music file 134, or the third music file 136 to play by the user of the computing device 102. As another example, the user can select one of the first music file 132, the second music file 134, or the third music file 136 to play by the user of the computing device 102.
Pat_GPT-J	The results panel 130 also displays the audio content of the third music file 136 for playback. The computing device 102 provides the audio content for playback. As described above, the computing device 102 may play back similar music files to the query music file 112, such as the first music file 132, the second music file 134, and the third music file 136. For example, the music search system 104 determines these music files to be similar to the query music file 112, and causes the computing device 102 to play the audio content of these music files.

Table 14: An example of patent specification generated by Patentformer, Pat_T5*, and Pat_GPT-J compared to the actual specification. The input text was embedded with rich context to generate the output specification.

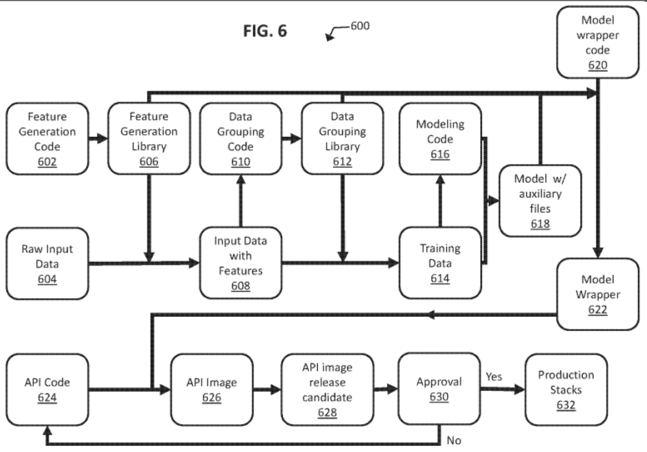
Input Figure	
FIG. 6  600 Model wrapper code 620 Feature Generation Code 602 Feature Generation Library 606 Data Grouping Code 610 Data Grouping Library 612 Modeling Code 616 Model w/ auxiliary files 618 Raw Input Data 604 Input Data with Features 608 Training Data 614 Model Wrapper 622 API Code 624 API Image 626 API image release candidate 628 Approval 630 Yes Production Stacks 632 No	
Input Text:	**<claim_feature>** 4. The method of claim 3, wherein the model wrapper is deployed only after the approval of the model wrapper is received. **<claim_feature_context>** 3. The method of claim 1, further comprising: sending, by the one or more processors after the model wrapper is generated, the model wrapper and an approval request to a supervisor electronic device; and receiving, by the one or more processors from the supervisor electronic device, an approval of the model wrapper in response to the approval request. **<brief_draw_desc>** FIG. 6 is a block diagram illustrating a process for automating deployment of a predictive model in accordance with one or more embodiments of the present disclosure. **<prev_para_num>** 75 **<prev_para>** As described herein with respect to various embodiments, after the model wrapper is generated and/or updated, any change to any of the feature generation code, the data grouping code, and/or the modeling code does not affect or change the model wrapper without reapplying the model wrapper code to any changed code. As also described herein, an approval request may be sent to a supervisor electronic device after the model wrapper is generated. An approval may then be received from the supervisor electronic device to approve of the model wrapper in response to the approval request. In various embodiments, the model wrapper may be deployed only after the approval of the model wrapper is received. In various embodiments, after the approval is received, the model wrapper may be stored on a server system such that the model wrapper is made available to receive incoming API calls. **<comp_name>** API code **<comp_num>** 624 **<comp_name>** API image **<comp_num>** 626 **<para_num>** 76 **<fig_num>** 6 **<spec>**
Actual	In various embodiments, the deployment of the model wrapper may include sending the model wrapper to a model wrapper repository and applying an API code, such as the API code 624 to the model wrapper to generate an API image configured to send and receive API calls. The API image may be, for example, the API image 626 of FIG. 6.
Pat_T5*	In some embodiments, after the approval is received, the model wrapper may be deployed. For example, after the approval is received, the model wrapper may be stored on a server system such that the model wrapper is made available to receive incoming API calls. In various embodiments, after the model wrapper is deployed, an API call may be received from a client electronic device to deploy the model wrapper and the API call may result in the deployment of the model wrapper. In various embodiments, the model wrapper may be deployed by running the API code 624 on the server system and the API image 626 on the client electronic device, respectively.
Pat_GPT-J	In accordance with one or more embodiments, FIG. 6 may be a diagram of a process for automating deployment of a predictive model in accordance with one or more embodiments of the present disclosure. For instance, the API image 626 and the API code 624 may be a result of applying the model wrapper to various different portions of the underlying predictive model. As described herein, the API code 624 and the API image 626 may comprise portions of source code, which, when executed by one or more processors, cause the predictive model to be implemented on a computing system. In some embodiments, the API image 626 and the API code 624 may be implemented as web services and may be accessible by a user or a developer via a network. In other embodiments, the API image 626 and the API code 624 may be implemented in any manner and may be executed by the computing system. Further, the API image 626 and the API code 624 may be updated after the model wrapper is generated and/or updated.

Table 15: An example of patent specification generated by Patentformer, Pat_T5*, and Pat_GPT-J compared to the actual specification. The input text was embedded with rich context to generate the output specification.