World Patent Information 62 (2020) 101983

Contents lists available at ScienceDirect

World Patent Information

journal homepage: http://www.elsevier.com/locate/worpatin

Patent claim generation by fine-tuning OpenAI GPT-2

Jieh-Sheng Lee *, Jieh Hsiang

Department of Computer Science and Information Engineering, National Taiwan University, Taiwan

A B S T R A C T

In this work, we focus on fine-tuning an OpenAI GPT-2 pre-trained model for generating patent claims. GPT-2 has demonstrated impressive efficacy of pre-trained language models on various tasks, particularly coherent text generation. Patent claim language itself has rarely been explored in the past and poses a unique challenge. We are motivated to generate coherent patent claims automatically so that augmented inventing might be viable someday. In our implementation, we identified a unique language structure in patent claims and leveraged its implicit human annotations. We investigated the fine-tuning process by probing the first 100 steps and observing the generated text at each step. Based on both conditional and unconditional random sampling, we analyze the overall quality of generated patent claims. Our contributions include: (1) being the first to generate patent claims by machines and being the first to apply GPT-2 to patent claim generation, (2) providing various experiment results for qualitative analysis and future research, (3) proposing a new sampling approach for text generation, and (4) publishing our fine-tuned GPT-2 model and sample code for future researchers to run on Colab.

1. Introduction

GPT2. Deep learning and pre-training models have demonstrated excellent results in several language tasks recently. Particularly, fine-tuning the pre-trained models such as ELMo (Embeddings from Language Models) [1], OpenAI GPT (Generative Pre-Training) [2], GPT-2 [3] and BERT (Bidirectional Encoder Representations from Transformers) [4] has become the best practice for state-of-the-art results. GPT-2 is the successor to GPT. Although both GPT-2 and BERT are capable of text generation, Wang and Cho [5] found that GPT-2 generations are of better quality. In fact, GPT-2 is claimed to be so powerful that the risk of its malicious use is high. For this reason, OpenAI decided to keep its largest model (1.5B parameters) closed so that there is more time to discuss its ramifications.

Effectiveness. In this work, we generated patent claims by using OpenAI GPT-2 source code[1] and fine-tuning the released model of medium size (355M). Overall we are impressed by how coherent and complicate the generated patent claims could be, although not all text are generated equally in terms of quality. We are also surprised by how few training steps (one step is one single batch/gradient update) were necessary for GPT-2 to generate the first text that looks like a patent claim. It is a matter of time that the largest and more powerful model will be released to the public. Therefore, it is better to start early and assess its impact on patent research.

Limitations. The objective of this work is to provide initial proof of concept. The limitations of this work are: (1) The coherency of text generation is on the surface form which is provided by GPT-2. How meaningful the generated claims are is a topic requiring both qualitative and quantitative analysis at a large scale in the future. (2) The training data is randomly sampled. We conjecture that selecting the training data in a specific way may produce more meaningful results beyond surface form. For example, if the training dataset is specific to electronics, it is less likely for the model to generate something irrelevant to electronics. We leave this hypothesis to the future.

2. Related work

Patent field. In the patent field, Aristodemou et al. [7] reviewed 57 recent articles on the use of artificial intelligence methods, machine learning, and deep learning approaches for analyzing intellectual property data. The analysis is further divided into four main categories: knowledge management, technology management, economic value, and extraction of information. Lupu et al. [8] pointed out that, among patent-related applications, modern neural networks are applied for machine translation primarily, and there is a wide open field of opportunities for other tasks such as patent analysis, patent valuation and patent classification. It was also anticipated that the remarkable success of deep learning will certainly be tested on patent data someday.

CS field. In the computer science field, NLP (Natural Language Processing) turns text into structured data, and NLG (Natural Language Generation) turns structured data back to text. Recently, transfer learning based on Transformer models [9], such as GPT, BERT, and

* Corresponding author.Computer Science and Information Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei, 10617, Taiwan.
E-mail address: d04922013@ntu.edu.tw (J.-S. Lee).

[1] https://github.com/openai/gpt-2

https://doi.org/10.1016/j.wpi.2020.101983
Received 29 June 2019; Received in revised form 9 January 2020; Accepted 15 June 2020
Available online 19 August 2020

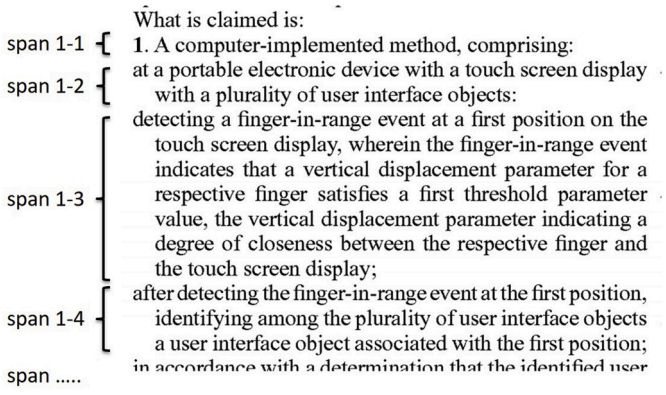

Fig. 1. Claim spans of US9229634B2.

GPT-2, outperformed significantly on various tasks after using pre-trained language models on large-scale corpora. The two-stage framework (pre-training & fine-tuning) is so effective that it is claimed as the arrival of the "ImageNet moment for NLP."[2] We observed that the success of Deep Learning in the NLP field has also spread to the NLG field. In this work, we are motivated to apply the latest NLG techniques to the patent field. We see it as an opportunity for patent professionals to generate or extract valuable information from patent data.

3. Data

Size. Our training dataset contains 555,890 patent claims of the granted U.S. utility patents in 2013. All of the claims are the first and independent claims. How to train GPT-2 with dependent claims and other independent claims is a topic for future research. We prepared our data from two perspectives: span-based and SQL-based. The former is about splitting a patent claim into shorter text spans. It makes claims easier to comprehend. The latter is about sharing SQL statements for future researchers, instead of sharing conventional raw data. The stages of our data pipeline include (1) raw data collection, (2) claim span identification and tagging, and (3) data encoding for GPT-2. In the rest of the section, we first explain our two perspectives, then the data pipeline.

3.1. Span-based

Claim spans. Patent claims are longer than ordinary sentences in NLP field. We observed that a lengthy patent claim is usually decomposed into multiple claim spans. A *claim span* is a segment of the claim text. We also observed that segmentations exist in human-curated claim text already. The separation of lines in official patent documents is an implicit segmentation. For example, as shown in Fig. 1. the first claim of the US9229634B2 patent is decomposed into several claim spans.

Granularity. The purpose of claim spans is two-folded. First, we measure how fast GPT-2 learns from patent corpus by observing how frequent such claim spans are generated. Second, a claim span is a reasonable approximation to an inventive step, component, or element for humans to comprehend. In contrast, the granularity of words or phrases in a patent claim is too fine. The granularity of the whole claim is too coarse. A claim span is a relatively suitable unit of inventive thought. Such segmentation of claim spans is rich human annotation and was probably never exploited in literatures.

3.2. SQL-based

BigQuery. Although raw patent data is available on the USPTO Open Data Portal,[3] we found it advantageous to leverage the Google Patents Public Datasets on BigQuery[4] at a higher level. Compared with conventional raw data, two advantages of using SQL statements are: (1) separation of concerns, i.e., the fetching and processing of raw data could be separated, and (2) clarity and flexibility, i.e., SQL statements are precise and easier to customize by different criteria. Usually, if the fetched data in a dataset has been processed and specific to a problem, it is often harder for other researchers to reuse the data for different situations or processing requirements. The SQL statement for our training dataset is listed in Appendix A.

3.3. Data pipeline & special tags

Stages. There are three stages in our data pipeline. The first stage is the raw data collection based on SQL statements. The second stage is to split patent claim text into claim spans. The third stage is to encode the claim spans in the required format for GPT-2 to digest. The first stage is simple. The second and the third stages have more implementation details as below.

Heuristic-based. In the second stage, we implemented the span segmentation on a heuristic basis. As mentioned, the official patent documents contain human-curated line breaks in patent claims. Our heuristic-based implementation is based on the observation that line breaks are often preceded by punctuation. The punctuation mark is either comma or semicolon most of the time. It is conventional to separate a patent claim in this way when filing a patent application. If not, the patent claim is likely to be in such format by human curation before granted.

Span segmentation. The punctuation marks alone are not sufficient to split a claim into spans, however. Comma and semicolon appear elsewhere too. Without extra information, it is hard to tell whether a comma acts as a span separator or just an ordinary mark in a claim span. We observed that, in the queried data from the BigQuery, the character return (i.e., line break) between lines was omitted. Combining the omission of character return and the specific punctuation marks, it is feasible to implement the span segmentation we need. For example, the character after a comma and semicolon should be a space character, if the punctuation mark means to be an ordinary mark instead of a line break. If the character after a comma or semicolon is not a space character, the punctuation mark is presumed to mean a line break. Such a heuristic span segmentation may be not perfect, but it is sufficient to split many patent claims into spans in an approximate sense.

Special tags. After identifying claim spans, we add a special tag "@@@" as a span separator to each span except for the last span. We follow Woolf's code[5] to add "<|startoftext|>" to the beginning of a patent claim and "<|endoftext|>" to the end. In this way we prepared our training dataset in a specific format. At the inference stage later, we can also identify the beginning and the end of a patent claim in all generated text. After identifying the patent claim, we can further split the generated patent claim into spans based on the span separator.

Encoded. The third stage of our data pipeline is straightforward. We use the encode function in the GPT-2 code and transform text data into a compressed numpy format (*.*npz*). The format is ready for training iterations and saving time. We shared our training data in both numpy format[6] and plain text format.[7] Future researchers can opt for preparing data from scratch (SQL) or reusing our formatted file for GPT-2.

[2] NLP's ImageNet Moment Has Arrived, http://ruder.io/nlp-imagenet/

[3] USPTO Open Data Portal, https://developer.uspto.gov/

[4] Google Patents Public Datasets on BigQuery, https://console.cloud.google.com/bigquery?p=patents-public-data

[5] gpt-2-simple, https://github.com/minimaxir/gpt-2-simple

[6] gpt2-claims-2013_for_345M.npz, https://data.mendeley.com/datasets/b8853hnj7b/draft?a=b99308ff-c24b-428c-96d7-d851962a2714

[7] gpt2-claims-2013.txt, https://data.mendeley.com/datasets/9dvny7cgcz/draft?a=6ba92bff-b464-4665-90c9-8e03f1ba4a13

4. Experimental setup

In this section, we explain the computing environment we worked on, the code base we derived from, and the model sizes of GPT-2 to consider.

4.1. Pre-trained models

Sizes. The four GPT-2 model sizes built by OpenAI are 124M, 355M, 774M, and 1.5B, in terms of the number of parameters in the neural network. Based on an interim update,[8] OpenAI released the larger 355M version as a staged release after the initial 124M model. At the same time, OpenAI shared the 774M and 1.5B versions with selected partners in the AI and security communities who are working to improve societal preparedness for large language models. At the beginning of our work, we found that the 124M model is sufficient for generating impressive results. Future researchers may start from the small model if computing resource is a constraint. In general, the larger the model is, the better the result becomes. Our experiments are based on the 355M model.

4.2. Colab & GitHub

Memory constraint. In terms of computing, we leverage Google Colab[9] for GPU and CPU. Colab is a Jupyter notebook environment that runs entirely in the cloud. The GPU available on Colab is NVIDIA Tesla T4 equipped with roughly 15 GB memory available, in our experiments. The memory size is sufficient for fine-tuning all layers of the small model (124M), but it is not sufficient for the medium model (355M). A known workaround is to use a memory-efficient gradient technique so that it works on Colab. We followed this approach based on Shepperd's repository[10] which is forked from OpenAI's repository. These repositories are the code bases for us to fine-tune the pre-trained model with patent claims.

TPU. It is noted that TPU (Tensor Processing Unit), more powerful than GPU, is also available on Colab. However, during our experiments, the public TensorFlow-based repositories work with GPU only. A popular PyTorch-based implementation of GPT-2[11] works with GPU only because the latest official release of PyTorch did not support TPU at the moment of our implementation. It is anticipated that both TensorFlow-based and PyTorch-based repositories would work on TPU smoothly soon.[12] If TPU is available, one of our follow-up tasks will be building a pre-trained model from scratch and from a patent corpus only. A patent-specific pre-trained model could be an essential building block for downstream patent tasks. For example, it would be interesting to know whether fine-tuning such a patent-specific model with CPC information can make patent classification tasks perform better or generate better patent claims. It would also be interesting to know whether fine-tuning the pre-trained with scientific papers can generate or discover relevant patent claims.

5. Experiments

Goals. We set four goals in our experiments. The first goal is to understand how fast the GPT-2 model adapts to patent claims. We approached this goal by observing the generated text in each step of early fine-tuning. Our second goal is to record the loss values during fine-tuning and observe their converging trend. Our third goal is to analyze the overall quality of the generated patent claims by unconditional random sampling. We devised a new sampling approach in pursuit of higher text quality. Our final goal is to compare the generated patent claims based on different text inputs for conditional random sampling. Readers interested in how to sample from language models can check the web page[13] for more details.

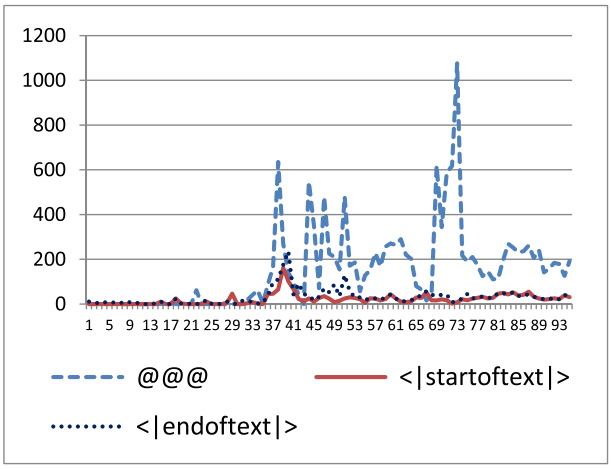

Fig. 2. Number of special tags in the first 100 steps of fine-tuning.

5.1. How fast GPT-2 adapts

Unreasonably fast. We measured how fast the GPT-2 adapts to patent claims by counting the occurrences of our special tags ("<|startoftext|>", "<|endoftext|>" and "@@@"") in generated text. It is a reasonable expectation that GPT-2 can generate more patent-like text if the number of fine-tuning step increases. It is to our surprise how few steps are required to generate the first patent-like text. As early as at the 36th step, GPT-2 generated the following text, and the positions of special tags are also appropriate:

> <|startoftext|> A hand-held mobile device comprising: @@@ a second hand with an articulated index finger comprising a first-stage rotator, a second-stage rotator, said second-stage rotator being rotatable upward and downward along a first pivot point, the apparatus extending through said first pivot point, said second-stage rotator being rotatable downward and upward towards said first pivot point, @@@ said first rotator being rotatable on a first end of said first pivot point when the apparatus is in said first-stage rotator position to provide an at least a second-stage rotator that rotates around a first pivot point; @@@ further, the apparatus has a first pivoting axis that rotates the pivot point; @@@ and the second hand pivoting axis has a second-stage rotator that rotates around a second pivot point, the apparatus rotating the at least a first-stage rotator of said first pivot point when said second-stage rotator is in said first-stage rotator position to provide said at least a second-stage rotator that rotates around a second pivot point. <|endoftext|>

It is unknown to us how many or how few patents were included in the original 40G WebText that the GPT-2 pre-trained models were trained on. If there were many, it would still be surprising to see the effectiveness of fine-tuning at a very early stage. If there were only a few, the effectiveness would be unreasonable. We leave this doubt to be clarified in the future.

First 100 steps. For archiving details, we collected the generated

[8] Better Language Models and Their Implications, https://openai.com/blog/better-language-models/
[9] Google Colaboratory, https://colab.research.google.com
[10] https://github.com/nshepperd/gpt-2/blob/finetuning/src/memory_saving_gradients.py
[11] https://github.com/huggingface/transformers
[12] Check PyTorch Lightning for latest development, https://github.com/PyTorchLightning/pytorch-lightning
[13] https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277

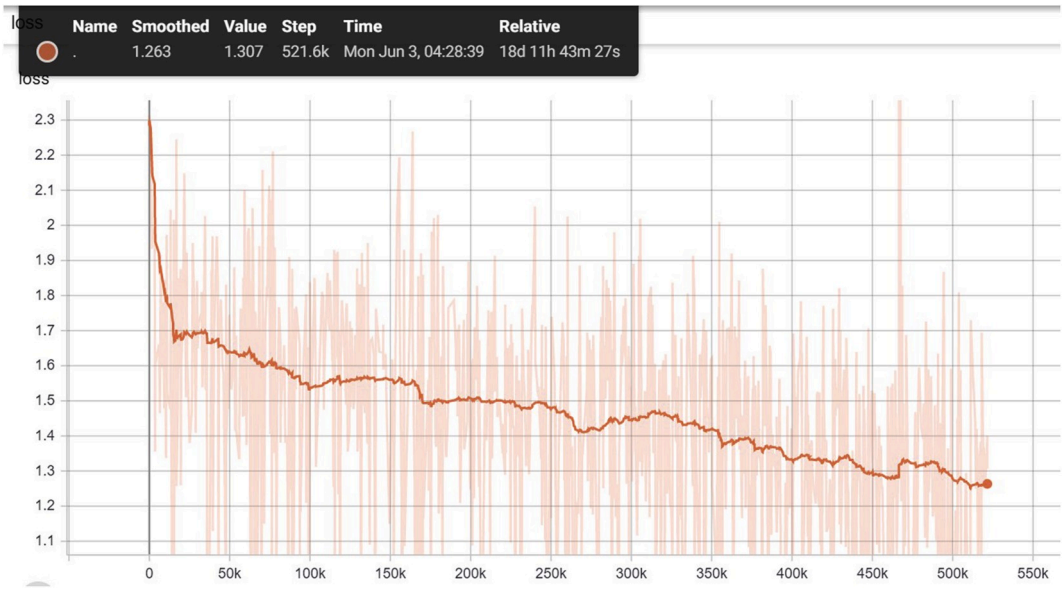

Fig. 3. Training loss in fine-tuning.

text in the first 100 steps of fine-tuning. The collection is made available online as research data for future study.[14] Statistically, the occurrences of the three special tags in the first 100 training steps are shown in Fig. 2. It is noted that not all of the occurrences make good sense. The chart is nevertheless a simple and intuitive way to suspect what might be happening in the black box.

Black box. A neural network is often a black box in the sense that even a simple neural network with a single hidden layer could be hard to understand. Interpreting black box models has been a challenge in Deep Learning for a long time. Compared with other neural network models, there is a chance that the attention mechanism in Transformer models may provide better interpretability. For example, Vig [10] presented an open-source tool for visualizing multi-head self-attention in Transformer-based language models. The tool is capable of visualizing attention at three levels of granularity (attention-head level, model level, and neuron level). It might provide more insights for understanding the first 100 steps from inside out. We leave this research topic to the future.

Hyperparameters. In our experiment, we built a baseline and kept the hyperparameters that are common in public repositories, specifically learning rate as *1e-4*, temperature as *1.0*, `top_k` as *40,* and batch size as *1*. We ever tried *1e-5* as the learning rate, but the convergence is too slow. Nothing like a patent claim was generated in the first 100 steps if the learning rate is *1e-5*.

5.2. Training loss in fine-tuning

Convergence. In this experiment, we used the same common hyperparameters but changed the learning rate from *1e-4* to *1e-5*. We expected a lower training loss. The convergence of the training losses is shown in Fig. 3. A lower learning rate leads to a slower convergence in general. Based on the trajectory, it is reasonable to us that the training loss is likely to decrease after more training steps. At which step it will become flat is unknown, however. How to use a different dataset to validate and prevent overfitting is also unknown. We leave this kind of topic to the future after having more computing resources.

5.3. Unconditional random sampling

Unconditional. In this experiment, we explored different approaches to unconditional sampling for patent claim generation. The original GPT-2 used the `top_k` random sampling with a default value *40*. It means sorting by probability and zero-ing out anything below the 40th token when sampling. Holtzman et al. [11] pointed out that a potential issue is the fixed threshold *k*, which might not be the best all the time. The quality of the generated text depends on the distribution of reasonable words in actual cases. If there are many words one could sample from reasonably, the number *k* might be too low. If there are only a few reasonable words to sample from, the number *k* might be too high. To solve this problem and have a dynamic number of samples, the authors proposed a *top_p* sampling called nucleus sampling. The essence of the *top_p* sampling is that the number of samples depends on zero-ing out anything below the cumulated distribution *p*.

Different cut-off. In this work, we propose a different cut-off approach called `dynamic_kp`. The cut-off threshold is based on the relative scale of probability compared with the probability of the top token. For example, in our experiment, we set the cut-off probability as *0.1* of the probability of the top token. We assumed that one order of magnitude is a significant boundary for cut-off and zero-ing out anything below. It has an effect to make the *k* value dynamic in `top_k` sampling and the *p* value dynamic in `top_p` sampling. We provide the essence of the `dynamic_kp` sampling as below. Whether the threshold *0.1* for `dynamic_kp` is the best is up to future experiments.

[14] https://github.com/WPI1/WPI_2019_58/blob/master/first_100_steps.txt

```
def dynamic_kp(logits, top_kp=0.1):
  k=100 # sufficient to identify cut-off

  probs_logits=tf.nn.softmax(logits)
  k_probs,_=tf.nn.top_k(probs_logits,
k=k)
  k_probs=tf.squeeze(k_probs)
  # top 100 probabilities

  probs_max=tf.reduce_max(k_probs)
  # max probability

  k_threshold=tf.multiply(probs_max,
top_kp)
  probs_mask=tf.to_int32(k_probs>=k_thre
shold)
  num_of_k=tf.count_nonzero(probs_mask,
dtype=tf.int32)
  # the number of tokens above cut-off

  # leverage original top_k code
  values,_=tf.nn.top_k(logits,
k=num_of_k)
  min_values=values[:,-1, tf.newaxis]
  return tf.where(logits < min_values,
tf.ones_like(logits, dtype=logits.dtype)
*-1e10,logits)
```

Comparison. For comparing different sampling results, we tried `dynamic_kp` *(0.1)*, `top_k` *(40)* and `top_p` *(0.9)* to generated 30 patent claims for each type. Without any cherry-picking, a complete list of the 90 patent claims is archived online as research data for review.[15] We conducted our qualitative analysis based on these 90 samples. Take the following as a positive example:

A method for a mobile station to communicate with a base station in a wireless communication system, the method comprising:

transmitting, to the base station, a first request to enter a high power state, wherein the first request is received according to an operating state;

receiving, from the base station, a second request to enter the high power state, wherein the second request is received according to a standby state;

determining whether the mobile station is in the standby state; and

entering the high power state upon determining that the mobile station is in the standby state.

Observation. We observed that the above generated claim seems coherent, making some practical sense, and having no obvious syntactical or semantical error. Overall we observed a significant number of generated claims having similar and acceptable quality. Among those claims, we also observed a plausible range of diversity and correct sequences of bullet items.

Negative. In contrast, the number of samples with more reduced quality is also significant. Notably, some claims are too long and hard to understand. The details of a claim may diverge and end up very far. Sometimes a term or phrase might be even repetitive or incorrect. How to fix these quality problems is a future research topic. For example, regarding the lengthy claims, it might be possible to split them into independent and dependent claims by a downstream task. Alternatively, such an issue might be mitigated (or become worse) after having dependent claims included in the training dataset. For brevity, we selected a few claim spans of more reduced quality as below. Interested readers could investigate the complete list online for details.

… …

wherein one or more of the control signal sets are used to generate a plurality of image display,

wherein one or more of the control signal sets are used to generate a plurality of new image display,

… …

receiving, at the first wireless access point, a third data packet from the second wireless access point, the third data packet sent from the second wireless access point.

… …

e) generating the sequence of data packets from the sequence of data packets based on the selected time for each data packet; and

f) transmitting the sequence of data packets, the sequence of data packets having a time and the stored time.

… …

5.4. Conditional random sampling

Conditional. In this experiment, we followed the same settings and tried conditional random sampling with two different inputs: (1) A deep learning method for patent analysis, (2) A deep learning method for drones. We generated 30 patent claims for each of the `dynamic_kp`, `top_k`, and `top_p` samplings. Without cherry-picking, the 90 generated patent claims for input (1) and (2) are archived online as research data, respectively.[16] Most of our observation is similar to our qualitative analysis on unconditional random sampling and omitted here. It is noted that the text generation quality depends on the input text, however. For example, if the input is longer, it is generally harder for all of the generated text to stay relevant to the input text. If the input is shorter and looks like the beginning of a claim, the text generation quality is usually better. We leave empirical study on quantitative analysis to the future and provide two positive examples to show some reasonable quality of text generation:

(1) A deep learning method for patent analysis, comprising the steps of:
    generating a plurality of patent scores for each patent of a plurality of patents in a training set of patent scores, each patent of the plurality of patents having a patent score for each of the plurality of patents in the training set;
    receiving a patent score for each of a plurality of patents in a training set of patents scores;
    generating a plurality of patent score differences for each of the plurality of patents, wherein a first patent score difference is generated for a first patent of the plurality of patents based on the received patent score of the first patent and a second patent score difference is generated for a second patent of the plurality of patents based on the received patent score of the second patent;
    comparing the first patent score difference and the second patent score difference; and

[15] https://github.com/WPI1/WPI_2019_58/blob/master/unconditional_sampling_1.txt

[16] https://github.com/WPI1/WPI_2019_58/blob/master/conditional_sampling_1.txt, https://github.com/WPI1/WPI_2019_58/blob/master/conditional_sampling_2.txt

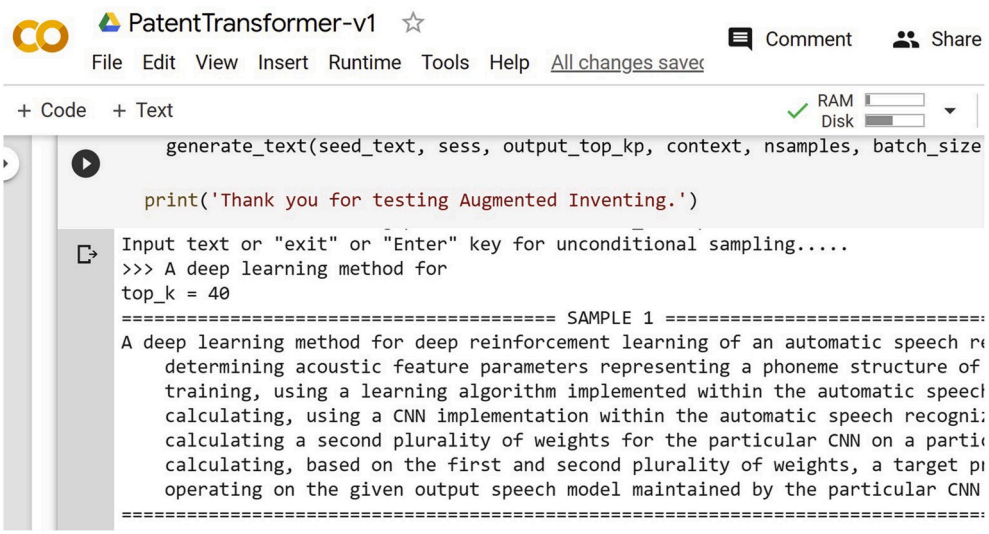

Fig. 4. Executable sample code on Colab.

generating a patent score for each of the plurality of patents based on the comparison.

(2) A deep learning method for drones, comprising the following steps:
   a. Creating an initial base grid and a final base grid by calculating a first total number of points and a first distance between the final base grid and the initial base grid;
   b. Setting up a first grid with a plurality of cells;
   c. Setting up a second grid with a plurality of cells;
   d. Setting up a third grid with a plurality of cells, wherein each cell of the second grid is connected to each cell of the third grid;
   e. Calculating a plurality of total distance durations for each cell in the second grid and the third grid;
   f. Calculating a plurality of total distance durations for each cell in the first grid and the second grid; and
   g. Calculating a plurality of total distance durations for each cell in the final grid and the first grid.

Auto complete. These two patent claims are very different, even though the majority of the input characters are the same. We observed that the possible details of "patent analysis" and "drones" were generated respectively with acceptable quality. It could be noted that the length of the input is short. The length of the output is comparatively long. Therefore, we contemplate on an "auto complete" use case in which, if an inventor is exploring new ideas and has no whole picture in mind yet, claim generation like this may be a way of augmented inventing. If the speed of GPT-2 inference is fast enough in the future, it should be possible to build an interactive patent drafting assistant. The assistant can suggest next terms, phrases, claim spans or even new ideas. This may open a new window for both qualitative and quantitative analysis on patent claim generation, too. For example, by measuring the gap between the user's actual next word and the probability distribution of candidate words in GPT-2, it is possible to measure the accuracy of inferencing. It is also possible to compare the accuracy of different sampling approaches. Such user interaction and quantitative metrics may shed more light on understanding Transformer models deeper.

5.5. Sample code on colab

Enhancements. Our source code for generating patent claims is available on GitHub.[17] The code is executable on Colab, as shown in Fig. 4. Free GPU is available on Colab. The majority of the code is derived from the official GPT-2 repository by OpenAI. The primary enhancements we made are: (1) loading the model we fine-tuned with patents, (2) generating text based on our new dynamic_kp sampling, in addition to the original `top_k` and `top_p` provided in the GPT-2 repository, and (3) detecting the end of a generated patent claim and splitting the claim into spans.

Usage. When executed, the prompt can accept the "Enter" key as empty input and generate text as unconditional sampling. For conditional sampling, one can input a few words as seed text. Empirically we found the quality of text generation will decrease if the seed text is more prolonged and unlike a patent claim. At the end of the test, one can type "exit" to end the program.

Variables. In the source code, we set "`nsamples = 1`" to run one iteration for each sampling algorithm (`top_k = 40`, `top_p = 0.9`, `top_kp = 0.1`). The "`temperature = 1`" is the default setting in GPT-2 code which controls the diversity of text generation. If the value is too high, the coherency will break. If the value is too low, the generated text may look similar or even the same. These variables are entry points for interested readers to experiment further.

6. Looking forward

Transformer-based models are at the early stage of development in the Deep Learning field. It will not be surprising if the next version of GPT-2 or BERT sets a new state of the art or the model size increases further. In this section, we look forward to briefly some recent efforts on Transformer-based models and their implications on patent claim generation in the future.

Supervised knowledge. First, patent classification is a kind of supervised knowledge that can be leveraged. Training a pre-trained Transformer model with supervised knowledge may outperform the original model. For example, Li et al. [12] investigated a transferable BERT training framework, which can transfer not only general language knowledge from large-scale unlabeled data but also specific kinds of knowledge from various related supervised tasks, such as next action prediction and sentiment classification. We expect that such a three-stage approach can generate better patent claims if the patent classification information can be learned into the model.

Multilingual. Second, patent claim generation in languages other

[17] https://github.com/jiehsheng/PatentTransformer

than English is another line of work. One possibility is to fine-tune a pre-trained model in a different language. Another possibility is to fine-tune a pre-trained multilingual model. The latter is compelling because Pires et al. [13] showed that a multilingual BERT model could perform cross-lingual generalization well. This means that the annotations in one language can be used to fine-tune the model for another language. We conjecture that the supervised knowledge, such as patent classification in other languages, can make multilingual patent claim generation more effective.

Data-driven. Third, a task in our plan is to train a larger model from scratch with a patent corpus only. Another planned task is to explore the possibility of better text generation by a second fine-tuning with a domain-specific dataset. Neural network models are data-driven. The model performance may come from either the algorithm inside the model or the data for training the model. Taking a data perspective seems less explored than an algorithm perspective.

7. Conclusion

The emergence of Transformer models such as GPT-2 is a paradigm shift and a tremendous opportunity for researchers in the patent field. In this work, we took a span-based approach and demonstrated the possibility of augmented inventing based on the text generation capability provided by the GPT-2 model. We released both of our fine-tuned 355M model which is specific to patents and our executable code on Colab. Hundreds of generated patent claims are published as research data on GitHub too. Our experiments show that the generated patent claims could be coherent on the surface form. Beyond the surface form, a more profound challenge is how to measure the semantic quality of text generation and generate better patent text. In summary, the proof of concept in this work is our first step toward the envisioned “auto complete” use case for Augmented Inventing.

CRediT authorship contribution statement

Jieh-Sheng Lee: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Jieh Hsiang:** Supervision, Writing - review & editing.

Acknowledgments

The research reported in this manuscript has been funded by the Ministry of Science and Technology (MOST) in Taiwan (Project: 106-2221-E-002-207-MY2).

Appendix A

- The following SQL selects the first claims of all US utility patents in 2013 and aggregates the CPC codes at subclass level (data source: Google Patents Public Datasets on BigQuery):

```
SELECT STRING_AGG (distinct t2. group_id order by t2. group_id) AS cpc_ids, t1.id, t1.date, text.
FROM ‘patents-public-data.patentsview.patent’ t1, ‘patents-public-data.patentsview.cpc_current’ t2,
‘patents-public-data.patentsview.claim’ t3
where t1.id = t2.patent_id
and t1.id = t3.patent_id
and timestamp (t1.date) ≥ timestamp (‘2013-01-01′)
and timestamp (t1.date) ≤ timestamp (‘2013-12-31′)
and t3.sequence = ’1’.
and t1.type = ’utility’.
group by t1.id, t1.date, t3.text.
```

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.wpi.2020.101983.

References

[1] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep Contextualized Word Representations, in: Proc. 2018 Conf. North American Chapter Assoc. Comput. Linguist. Hum. Lang. Technol., 1, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 2227–2237. http://doi.org/10.18653/v1/N18-1202.

[2] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-training (Unpublished Manuscript), 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners. https://www.ceid.upatras.gr/webpages/faculty/zaro/teaching/alg-ds/PRESENTATIONS/PAPERS/2019-Radford-et-al_Language-Models-Are-Unsupervised-Multitask-Learners.pdf, 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional Transformers for language understanding, in: Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, vol. 1, 2019, pp. 4171–4186 (Long Short Pap. https://aclweb.org/anthology/papers/N/N19/N19-1423/.

[5] A. Wang, K. Cho, BERT has a mouth, and it must speak: BERT as a Markov random field language odel. http://arxiv.org/abs/1902.04094, 2019.

[7] L. Aristodemou, F. Tietze, The state-of-the-art on Intellectual Property Analytics (IPA): a literature review on artificial intelligence, machine learning and deep learning methods for analysing intellectual property (IP) data, World Patent Inf. 55 (2018) 37–51, https://doi.org/10.1016/j.wpi.2018.07.002.

[8] M. Lupu, Information retrieval, machine learning, and Natural Language Processing for intellectual property information, World Patent Inf. 49 (2017) A1–A3, https://doi.org/10.1016/j.wpi.2017.06.002.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, In Advances in Neural Information Processing Systems, 2017. http://papers.nips.cc/paper/7181-attentionis-all-you-need.pdf.

[10] J. Vig, Visualizing Attention in Transformer-Based Language Representation Models, 2019. http://arxiv.org/abs/1904.02679. (Accessed 26 April 2019).

[11] A. Holtzman, J. Buys, M. Forbes, Y. Choi, The Curious Case of Neural Text Degeneration, 2019. http://arxiv.org/abs/1904.09751. (Accessed 21 May 2019).

[12] Z. Li, X. Ding, T. Liu, Story Ending Prediction by Transferable BERT, 2019. http://arxiv.org/abs/1905.07504. (Accessed 2 June 2019).

[13] T. Pires, E. Schlinger, D. Garrette, How Multilingual Is Multilingual BERT?, 2019. ArXiv1906.01502v1 [Cs], http://arxiv.org/abs/1906.01502v1. (Accessed 10 June 2019).

Jieh-Sheng Lee is currently a PhD candidate in the Department of Computer Science and Information Engineering at National Taiwan University and also an in-house patent attorney at Novatek Microelectronics Corp. His research focuses on applying Deep Learning to patents, particularly patent text generation based on latest NLP techniques. While in university, his team won the regional Championship of the ACM International Collegiate Programming Contest in Taiwan. His master thesis in Computer Science and Information Engineering won the First Prize of the Acer Dragon Thesis Award. The other master thesis in the Institute of Technology Law at National Chiao Tung University won the same First Prize again. He passed the USPTO bar exam in 2009 and has been licensed to practice law in New York since 2012.

Jieh Hsiang is a Distinguished Professor in Computer Science and Information Engineering at the National Taiwan University (NTU) and a Research Fellow of the Institute of Information Systems of the Academia Sinica. He is also the Director of the Research Center of Digital Humanities of NTU. He was the University Librarian of NTU between 2002 and 2008, and was a full professor in Computer Science at the State University of New York at Stony Brook before returning to Taiwan in 1993. He has authored over 150 research papers and several books, and has received a number of research awards, particularly one Test-of-Time Award from IEEE, two Outstanding Research Awards from the National Science Council of Taiwan, and one Academia/Industry Collaboration Award from the Ministry of Education. His service as the University Librarian also resulted in two Distinguished Service Awards, one from the Taiwanese Library Association and one from the National Taiwan University.