

IPR2024-01234

Declaration of Michael Pazzani

Filed on behalf of FriendlyAI, Inc.

By:

JENNIFER R. BUSH, Reg. No 50,784
JAE WON SONG, Reg. No. 59,070
JAE YEON BAEK, Reg. No. 78,258
MICHAEL J. SACKSTEDER (*pro hac vice*)
FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
Telephone: 650.988.8500
Facsimile: 650.938.5200

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

HUGGING FACE, INC.
Petitioner,

v.

FRIENDLIAI INC.,
Patent Owner.

Case No. IPR2024-01234
Patent No. 11,442,775

DECLARATION OF MICHAEL PAZZANI

I, Michael Pazzani, Ph.D., declare as follows:

I. INTRODUCTION

1. I am submitting this declaration in connection with the Patent Owner Preliminary Response of FriendliAI Inc. (“FriendliAI”; “Patent Owner”) to a Petition for *Inter Partes* Review (“Petition”) filed by Hugging Face, Inc. (“Hugging Face”; “Petitioner”). challenging claims 1-18 of U.S. Patent No. 11,442,775 (the “’775 patent”; Exhibit 1001) in IPR2024-01234 before the Patent Trial and Appeal Board of the United States Patent Office (the “Board”). I have been asked by FriendliAI to provide information and opinions regarding the level of skill of a person of skill in the art (“POSITA”) for the invention claimed in the ’775 patent, the general knowledge of such a person at the time of the filing of the application for the ’775 patent, the scope and content of the documents described by the Petition, technical information relating to the ’775 patent, and whether the claimed invention in the ’775 patent is directed to an inventive concept or would have been obvious to a POSITA in view of the grounds asserted by the Petitioner.

II. QUALIFICATIONS

2. I am presently an Affiliated Scientist having retired on October 4, 2024 from a Distinguished Principal Scientist position at the Information Sciences Institute at University of Southern California. I have worked in this role since 2021. From 2019 to 2022, I was a Distinguished Scientist at the Halıcıoğlu Data

Science Institute at the University of California, San Diego. From 2012 to 2019, I was a professor of computer science and engineering and Vice Chancellor for Research and Economic Development at the University of California, Riverside. From 2006 to 2012, I was a distinguished professor of Computer Science and Vice President for Research and Economic Development at Rutgers University. From 2002 to 2006, I worked as the Division Director of Information and Intelligent Systems at the National Science Foundation (“NSF”), a federal agency that funds fundamental research at universities. This division, with an annual budget of approximately \$250M, provided funding for research on topics that included Internet Search, Natural Language Processing, Artificial Intelligence, Digital Libraries, Machine Learning, Robotics, Computer Vision, and Human-Computer Interaction.

3. In 2000, while on leave from UC Irvine, I was CEO and co-founder of AdaptiveInfo, a company that personalized search and delivery of Internet news, classified ads, and restaurant listings for mobile devices such as cell phones. AdaptiveInfo was sold the company to Versifi in 2001. From 1988 to 2005, I was a professor of Information and Computer Science at the University of California, Irvine where I taught courses in Artificial Intelligence, Information Retrieval, Search Engines, and Machine Learning. I conducted research and supervised

graduate students on projects that included Internet Search, Personalization and Recommendation Systems, and Machine Learning.

4. My academic credentials include two degrees from the University of Connecticut: (1) a Bachelor of Science degree in Computer Engineering earned in 1980; and (2) a Master of Science degree in Computer Engineering earned in 1981. I earned a Doctor of Philosophy (Ph.D.) in Computer Science from the University of California, Los Angeles, in 1988.

5. I am an author or co-author of 48 journal articles, 83 conference papers, 14 book chapters and one book. I have also presented at approximately 32 workshops and edited a book.

6. I have extensive experience in the areas of search engines, machine learning, document classification and information retrieval, image classification, deep learning, and explainable AI: (a) creating and teaching a graduate course at UC Irvine— Information Retrieval, Filtering, and Classification (ICS 207); (b) teaching tutorials on document classification, machine learning and search at IJCAI, AAAI, and IUI conferences; (c) running a company that handled news, classified, and restaurant search for the Los Angeles Times and San Diego Union Tribune websites; (d) directing the Information and Intelligent Systems Division at the NSF, which provided funding for digital libraries, internet search, databases and machine research; (e) receiving a Classical Paper award from AAAI on

personalization in web search; (f) creating a specialized search engine that classified proposals for the NSF to find reviewers for research proposals; (g) training and advising multiple PhD graduates who have worked on commercial search engines at Yahoo, Google, and Yandex; and (h) serving on the Technical Advisory Board on Internet Search for Microsoft.

7. I received many honors, awards, research grants and gifts. In 2014, I was awarded the AAAI Classic Paper Award for a 1996 publication “for significant contributions to the field of personalizing Internet content and learning user profiles.” See <http://www.aaai.org/Awards/classic.php>. In 2003, I was recognized as a Fellow of the Association for the Advancement of Artificial Intelligence (“AAAI”) “for significant contributions to the development and understanding of machine learning algorithms and their relationship to human learning.” See <http://www.aaai.org/Awards/fellows-list.php>. I have received 37 grants worth roughly \$16.9 million total dollars. From 2019-2021, I served on The Defense Science Board, which provides the Secretary of Defense with independent advice and recommendations on scientific, technical, and other matters of special interest to the US Department of Defense.

8. My full *curriculum vitae* is attached as Exhibit A to this Declaration, which includes my detailed employment background, professional experience,

publications, honors and awards, and other qualifications to render an expert opinion.

III. DOCUMENTS AND MATERIALS REVIEWED

9. I have reviewed the challenged patent, U.S. Patent No. 11,442,775 (Ex. 1001). I have also reviewed the Petition, the Exhibits to the Petition (Exs. 1001-1022), including the Declaration of Ghaith Hammouri, Ph.D. (Ex. 1002). I have also reviewed all of the exhibits cited in the POPR and herein (Exs. 2002-2009, 2011-2019). In the preparation of this declaration, I relied on the relevant materials that Petitioner's expert relied upon and cited in his declaration, my knowledge of computer science, and my professional experience, both academically and in industry.

10. I am being compensated for my time. My compensation is not contingent upon the outcome of this *inter partes* review.

IV. LEGAL STANDARDS

11. This section below sets forth certain legal standards that have been provided to me by FriendlyAI's attorneys. I am not an attorney, and I am relying on these legal standards only to guide my analysis.

A. Scope of *Inter Partes* Review

12. I understand that, in this case, the grounds for institution include only allegations that the challenged claims are anticipated, obvious, and directed to non-patent eligible subject matter.

B. Claim Construction

13. I understand that “[i]n a *inter partes* review proceeding, a claim of a patent...shall be construed using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b), including construing the claim in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. §42.200(b).

14. Except as discussed below, for purposes of my analysis in this *inter partes* review, I applied the ordinary and customary meanings as would have been understood by a POSITA in view of the specification of the patent and the prosecution history.

15. I understand that a patent may include two types of claims, independent claims and dependent claims. I understand that an independent claim stands alone and includes only the limitations it recites. I understand that a dependent claim depends from an independent claim or another dependent claim. I

understand that a dependent claim includes all the limitations that it recites in addition to the limitations recited in the claim (or claims) from which it depends.

16. In comparing the challenged claims to the prior art, I have carefully considered the patent and its file history in light of the understanding of a person of ordinary skill in the art at the time of the invention.

17. I understand that to determine how a person of ordinary skill would have understood a claim term, one should look to sources available at the time of the alleged invention that show what a POSITA would have understood the disputed claim language to mean. It is my understanding that this may include what is called “intrinsic” evidence as well as “extrinsic” evidence.

18. I understand that, in construing a claim term, one should primarily rely on intrinsic patent evidence, which includes the words of the claims themselves, the remainder of the patent specification, and the prosecution history. I understand that extrinsic evidence, which is evidence external to the patent and the prosecution history, may also be useful in interpreting the patent claims when the intrinsic evidence itself is insufficient. I understand that extrinsic evidence may include dictionaries, principles, concepts, terms, and other resources available to those of skill in the art at the time of the invention.

19. I understand that the claims of the patent define the scope of the rights conferred by the patent. I understand that because the claims point out and

distinctly claim the subject matter which the inventors regard as their invention, claim construction analysis must begin with and is focused on the claim language itself. I understand that the context of the term within the claim as well as other claims of the patent can inform the meaning of a claim term. For example, because claim terms are normally used consistently throughout the patent, how a term is used in one claim can often inform the meaning of the same term in other claims. Differences among claims or claim terms can also be a useful guide in understanding the meaning of particular claim terms.

20. I understand that a claim term should be construed not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the entire specification. I understand that because the specification is a primary basis for construing the claims, a correct construction must align with the specification.

21. With regard to extrinsic evidence, I understand that all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises, can also be considered. For example, technical dictionaries may indicate how one of skill in the art used or understood the claim terms.

C. Obviousness

22. I understand that a claim may be invalid as obvious if the differences between the claimed subject matter and the prior art are such that the claimed subject matter would have been obvious to a person of ordinary skill in the pertinent art at the time of the invention. A claimed invention may also be invalid as obvious if a single reference or two or more references combined together teach or suggest all of the elements of the claimed invention. Obviousness is determined by looking at the claimed subject matter as a whole through the eyes of one of ordinary skill in the art at the time the claimed invention was made. Therefore, a claim is not obvious if the combined references do not teach or suggest at least one element of the claim. Further, a claim is not obvious if there is no suggestion or motivation to combine the references or a lack of some articulated reasoning with a rational underpinning to support the combination of references.

23. A claim is not proved obvious merely by demonstrating that each of the elements was independently known in the prior art. Most, if not all, inventions rely on building blocks long since uncovered, and claimed discoveries almost of necessity will likely be combinations of what is already known. It is important to identify whether a reason existed at the time of the invention that would have prompted a POSITA in the relevant field to combine the known elements in the way the claimed invention does. In assessing obviousness, one should consider the

distortion caused by hindsight bias, guard against slipping into the use of hindsight, be cautious of arguments that rely upon after-the-fact reasoning, and avoid the temptation to read into the prior art the teachings of the invention at issue.

24. The following factors must be evaluated to determine whether a petitioner has met its burden of proof that a claimed invention is obvious:

- a) the scope and content of the prior art relied upon by the petitioner;
- b) the difference or differences, if any, between each claim of the patent and the prior art;
- c) the level of ordinary skill in the art at the time the invention of the patent was made; and
- d) various secondary considerations of obviousness.

25. Secondary considerations include the invention's commercial success, commercial acquiescence (*i.e.*, licensing), a long felt but unresolved need, the failure of others, skepticism by experts, praise by others, teaching away by others, recognition of a problem, laudatory statements by the alleged infringer, and copying of the invention by competitors.

26. Dependent claims include all the limitations of the independent claim they are based on. Therefore, if a reference or combination of references fails to properly disclose, teach, or suggest a limitation of an independent claim, they also

fail to do so when that limitation is deemed incorporated into all the dependent claims. Thus, I have been informed and understand that if an independent claim is found invalid, every claim which depends from the independent claim is also invalid.

V. TECHNOLOGY BACKGROUND AND THE PATENT AT ISSUE

A. Field of the Invention

27. The relevant field of the invention of the '775 patent is “machine-learning *transformer* neural network models, and more particularly [] selective batching *for transformer models*.” Ex. 1001, 1:7-9 (emphasis added).

B. Level of Ordinary Skill in the Art at the Time of the Invention

28. Petitioner’s expert and the Petitioner propose that a POSITA of the '775 Patent would have “either: (1) a bachelor’s degree in electrical engineering, computer engineering, or computer science, with two to three years of work experience in machine learning; or (2) a master’s degree in electrical engineering, computer engineering, or computer science, with one year of work experience in machine learning.” Petition, 27 (citing Ex. 1002 ¶¶26-31).

29. I disagree to the extent that I believe a POSITA would have slightly more education and experience. In my opinion, for this patent, a POSITA would have at least a *master’s degree* in computer science, computer engineering, electrical engineering, or a related field in computing technology, and at least *two*

years of computer science (or related) experience with machine learning, or equivalent education, research experience, and knowledge. However, with more experience, less education may be needed, and vice versa.

30. I do not believe that the differences between Petitioner’s proposal and my own impact my analysis below. I have sufficient education and experience to qualify as a POSITA.

C. The ’775 Patent

31. The ’775 patent is directed to “machine-learning transformer neural network models, and more particularly [] selective batching for transformer models.” Ex. 1001, 1:7-9.

32. In the artificial intelligence (AI) computing field, the operation of machine-learning neural networks is computationally intensive. The ’775 patent is directed to improving the performance of transformer neural network models by selective batching, thus enabling the transformer to better “exploit the amount of parallel computation units in the hardware accelerators.” Ex. 1001, 1:40-42. Specifically, the ’775 patent applies iteration-level batching in AI transformer models. The patent explains, “at one or more iterations, the inference system can modify the batch being executed on the execution engine by adding new incoming requests to the batch or removing completed requests from the batch.” *Id.*, 2:62-64.

33. The claimed invention overcame then-existing problems in the field, including latency and low throughput, by batching and scheduling requests at an iteration-level. This action is known as “dynamic batching” or “continuous batching.” “[D]ynamic batching allows the serving system 435 to dynamically adjust batches that are processed on execution engines such that the hardware of the execution engine can be fully utilized.” Ex. 1001, 24:28-31. Some advantages of the invention are that “the response can be provided to the user faster,” it avoids the execution engine “being under-utilized,” and it achieves “higher processor utilization on the hardware accelerators.” *Id.*, 3:2-6, 5:30-32.

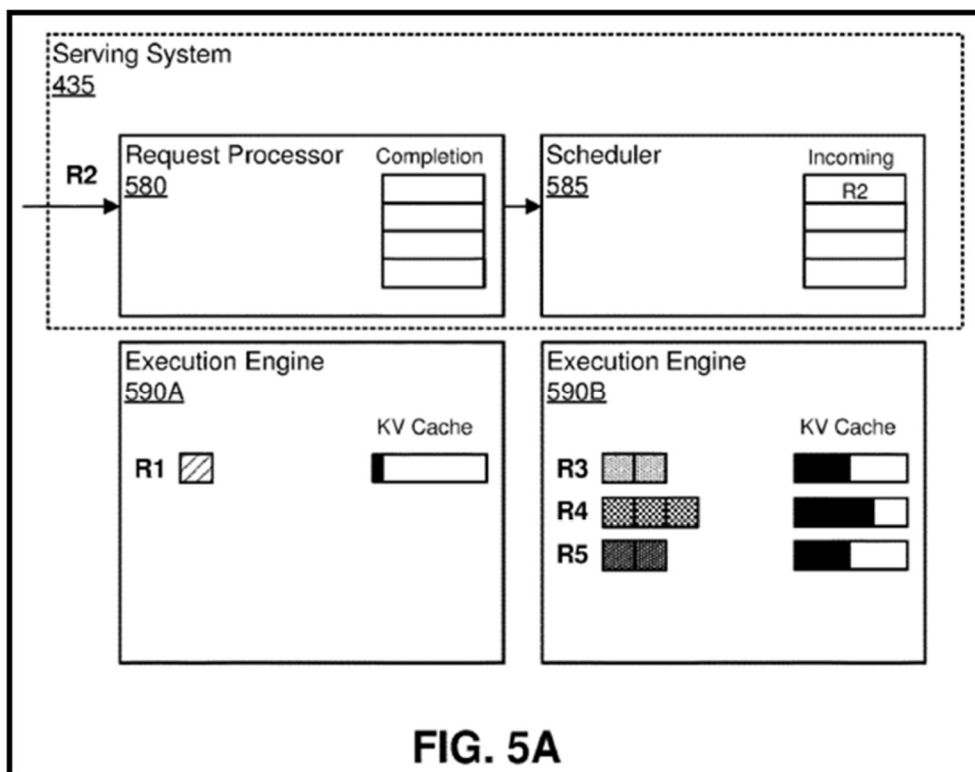
34. The '775 patent enables transformer batches to have variable input length, variable target length, and variable internal state length. Ex. 1001, 1:55-61. This is not possible in RNNs. In previous machine-learning transformer models, the model architecture “require[d] the length of inputs or the lengths of internal states to the transformer model be the same” across the batch. *Id.*, 21:63-22:1. With the '775 patent,

dynamic batching allows the serving system 435 to dynamically adjust batches that are processed on the execution engines such that the hardware of the execution engine can be fully utilized. This process was not available for certain methods of batching (e.g., method of FIGS. 2A-2B) for transformer models, because internal states, such as the key cache tensor and value cache tensor maintain

variable length per iteration in a transformer model, while other recurrent machine-learning models (e.g., RNNs) make use of state data that have the same lengths across a batch of requests.

Id., 24:28-38.

35. The system of the '775 patent is depicted in the following figure:



36. Ex. 1001, Fig. 5A. Fig. 5A depicts an embodiment of the patented invention, where the serving system 435 is comprised of the request processor 580 and the scheduler 585. Ex. 1001, 22:25-27. In the context of a text generation system, where a user may ask questions to the system and expect to receive answers to their queries, the user requests are depicted by the labels R1-R5. *Id.*, 22:28-38; 22:47-51. Requests are received by the serving system and sent to the

scheduler. *Id.* The scheduler then adds those requests to batches of requests, and schedules them for execution on an execution engine. *Id.*, 22:38-44. When memory becomes available, new requests may also be added to a batch to be executed with one or more of the previously executing requests. *Id.*, 23:26-29.

37. For example, claim 1 of the '775 patent recites claim language directed to iteration-level scheduling (bolded and italicized) below:

1[pre] 1. A method of *dynamically executing batches of requests* on one or more execution engines running a machine learning transformer model, comprising:

1[a] receiving, by a serving system, one or more requests for execution, the serving system including a scheduler and a machine-learning transformer model including at least a set of decoders;

1[b] *scheduling, by the scheduler, a batch of requests including the one or more requests for execution on an execution engine;*

1[c] *generating, by the execution engine, a first set of output tokens* by applying the transformer model to a first set of inputs for the batch of requests, wherein applying the transformer model comprises applying at least one batch operation to one or more input tensors associated with the batch of requests;

- 1[d] receiving, by a request processor, a new request from a client device, the new request including a sequence of input tokens;
- 1[e] *scheduling, by the scheduler, a second batch of requests additionally including the new request for execution on the execution engine*, the second batch of requests scheduled responsive to determining that the execution engine has memory available to execute the second batch of requests, wherein in a second set of inputs for the second batch of requests, a length of the sequence of input tokens for the new request is different from a length of an input for at least one request other than the new request; and
- 1[f] *generating, by the execution engine, a second set of output tokens* by applying the transformer model to the second set of inputs for the second batch.¹

D. Prosecution History of the '775 Patent

38. I have reviewed the prosecution history of the '775 Patent. The '775 Patent is entitled “Dynamic Batching for Inference System for Transformer-Based Generation Tasks” and was filed on December 3, 2021 as U.S. Patent Application

¹ Claim 10 recites similar substance in computer-readable medium format.

No. 17/542,193 (“the ’193 Application”). Ex. 1001. The ’193 Application is an original nonprovisional application. Ex. 1010, 5. The ’193 Application was originally filed with claims 1-18. Ex. 1010, 51-56. There is no terminal disclaimer associated with the ’775 patent. See Ex. 1001, cover.

39. On February 1, 2022, Patent Owner filed an Information Disclosure Statement which disclosed the Gao reference (Ex. 1004). Ex. 1010, 110. On February 9, 2022, the Examiner rejected claims 1-4, 7-13, and 16-18 under 35 U.S.C. §103 as being unpatentable over Aarts et al. (US 2021/0192314 A1) in view of Chung et al. (US 10,846,096 B1). *Id.*, 120.

40. The Examiner conducted an applicant interview on April 27, 2022, and the Aarts and Chung references were reconsidered. Ex. 1010, 187. Upon review, the Examiner agreed that both references apply specifically to RNNs and not to transformers. Ex. 1010, 189-191. Regarding Aarts, the Examiner stated, “Aarts also does not disclose batching *in the context of machine-learned transformer models*. In addition, Aarts also does not disclose applying a transformer model to a first batch and subsequently scheduling a second batch of requests...” (emphasis added). Ex. 1010, 191. Regarding Chung, the Examiner stated, “Chung does not describe batching *in the context of machine-learned transformer models*. In addition, Chung does not disclose applying a transformer model to a [first] batch of requests on a GPU...” (emphasis added). Ex. 1010, 189-

190. The Examiner concluded that Aarts and Chung (which relate to RNNs) can be differentiated from the '193 Application (which relates to transformers), and therefore, "It was agreed that the proposed claim amendment... would overcome the current prior art rejections." Ex. 1010, 177.

41. Patent Owner filed an Amendment to the claims that, *inter alia*, added text to claims 1 and 10: "wherein in a second set of inputs for the second batch of requests, a length of the sequence of input tokens for the new request is different from a length of an input for at least one request other than the new request." Ex. 1010, 181. The Examiner allowed the '193 Application on May 11, 2022. Ex. 1010, 195.

42. Different neural network architectures are used in artificial intelligence. Those relevant to the discussion here are transformers, which are articulated in the claims of the '775 patent. Also pertinent to Petitioner's analysis is a different class of neural network architectures called recurrent neural network architectures (RNNs), which are the focus of both the Gao and Katharopoulos references. I address each below.

VI. STATE OF THE ART

A. Recurrent Neural Networks (RNNs)

43. In sequential data modeling, given a user prompt, a trained model iteratively generates tokens that represent text units (e.g., words, sub-words) as a

response to the user prompt. Specifically, text units in the user prompt are converted into a sequence of input tokens. A token is a numerical representation of a text unit that allows machine learning (ML) and AI models to process the text units mathematically. For example, if a token represents a word, a trained model processes the sequence of input tokens to generate an output token representing the likely next word in the sequence. Ex. 2004, 137-138. The output token is then used as the input to the model in the next iteration and the next output token is generated. This process is repeated until a token signaling the “end” is generated as the output token.

44. RNNs are a class of neural network models that apply a trained set of parameters to input data in a recurrent manner to model and generate sequences of data (e.g., text, speech). The RNN includes connections that span adjacent iterations, introducing a notion of time to the model. Ex. 2002, §3. Although they date back to the 1980s, and the 1990s, use of RNNs became more practical as the availability of digital data and computing power and memory increased. *See* Ex. 2009, 5-11, Ex. 2011, 182-185.

45. The RNN receives inputs (e.g., tokens from user prompt or previously generated output tokens) and generates the next output token. Specifically, to generate the output tokens, RNNs generate a “hidden state” for each iteration that allows the model to maintain a memory state of the tokens that have been

processed so far in the sequence when generating an output token for an iteration.

The hidden state is iteratively updated across time, as a function of the inputs and prior hidden state, and acts as a compressed summary of the sequence processed so far. The length or dimensionality of the hidden state of an RNN is fixed over time.

46. A representative RNN architecture is shown in FIG. A below, where the t -th input token is represented as $x^{(t)}$, the t -th hidden state is represented as $h^{(t)}$, and the t -th output token is represented as $y^{(t)}$. Ex. 2002, §3; Ex. 2004, §7.3.

While there are variations of RNNs, the description herein focuses on operations of the “Seq2Seq” RNN model, as this is the type of RNN model that is referred to throughout the Petition. The Seq2Seq model includes an encoder RNN and a decoder RNN.

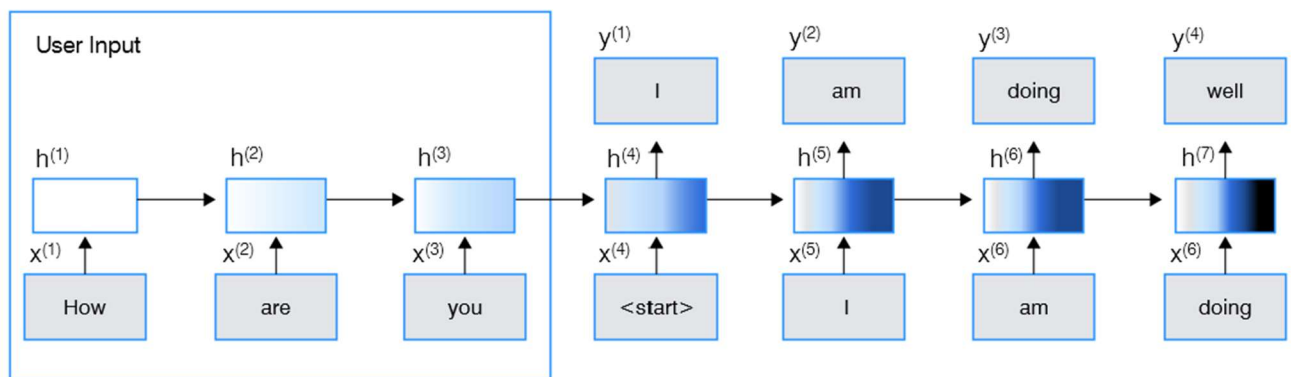


FIG. A. Overview of RNNs²

² FIGS. A-F were created by Patent Owner to illustrate RNN and transformer models and batching in each.

47. The encoder RNN during an “encoding phase” processes the tokens of the user prompt to generate a hidden state that incorporates the contextual information of the user prompt tokens that is later used to generate the first output token of the response. The hidden state $h^{(t)}$ is updated by combining the current input $x^{(t)}$ multiplied by a learned weight matrix and the previous hidden state $h^{(t-1)}$ multiplied by a learned weight matrix. Ex. 2002, §3; Ex. 2004, §7.3. For example, FIG. A illustrates the token for “How” is used to generate hidden state $h(1)$, the token for “are” and previous hidden state is used to generate hidden state $h(2)$, and the token for “you” and previous hidden state is used to generate hidden state $h(3)$.

48. The decoder RNN during a “decoding phase” takes the final hidden state from the encoder RNN and iteratively updates the hidden state while iteratively generating output tokens. The output token at each iteration is generated based on the current hidden state $h^{(t)}$. FIG. A illustrates the start token “<start>” and the previous hidden state from the encoder RNN is used to generate hidden state $h(4)$, which is used to generate output token “I.” The previous output token “I” is used as the input for the next iteration and the previous hidden state is used to generate hidden state $h(5)$, which is used to generate output token “am.” The token “am” is used as the input for the next iteration and the previous hidden state is used to generate hidden state $h(6)$, which is used to generate output token “doing.” The token “doing” is used as the input for the next iteration and previous

hidden state is used to generate hidden state $h(7)$, which is used to generate output token “well.”

49. However, RNNs suffer from inherent challenges. Ex. 2002, §3.2. As I discussed above, the size of the hidden state of an RNN is fixed over time. Ex. 2005, §2. The performance of the RNN becomes limited as the sequence gets longer, since the RNN layers compress the context of the sequence into the hidden state of fixed dimensionality. Ex. 2005, §2. This is shown in FIG. A via the diminishing size of the white and lighter portions within the hidden states over time. Another difficulty is that RNNs require sequential processing, where each iteration during training depends on the output token of the previous iteration. Thus, it is difficult to parallelize computation, slowing down training especially for longer sequences. Moreover, the training process of the RNN makes it difficult to learn long-term dependencies because of the way the architecture of the RNN is designed. *See* Ex. 2002 (§3.2).

50. One technology that bridged RNNs into the use of transformers, but is not discussed in the Petition, is LiSTM. *See, e.g.*, Ex. 2017.

B. Transformers

51. Transformers are a more recent, revolutionary class of models for sequential data modeling that were developed in 2017 by Vaswani, in part to address the shortcomings of RNNs. Ex. 1005, 1. A key component of the

transformer model is the attention mechanism. Ex. 2008, §2.4.1; *see generally* Ex. 2006. Given an input for a current iteration, the attention mechanism allows the transformer model to “attend” to different words of the sequence generated thus far when generating the next output token, so the significance of specific words in the sequence are weighed when understanding the text. Ex. 2008, §2.3.1. Transformers excel at processing sequences of data, such as text or speech, and transforming them into another sequence, like translating text into another language, answering questions, or summarizing information.

52. Transformer models model the contextual information for a sequence in an inherently different manner from RNNs. Specifically, as tokens are generated for a sequence, the contextual information for each token is encoded as “keys” and “values” in each attention layer. For a given input at a current iteration (e.g., tokens for a user prompt or a previously generated output token), a “query” is generated at an attention layer that is a transformation of the input, and embeds information on what the transformer model may want to pay attention to for generating the next output token. The attention layer also generates “keys” and “values” for each token in the sequence generated thus far. The “key” for a token in the sequence describes what the token is “offering” or when the token is important for the query. The “value” for a token in the sequence represents the content of that token within the generated context.

53. The attention output for an attention layer is a weighted sum of the “values” for the tokens of the sequence, each weighted by a respective importance as determined by the respective “key” for the token with the “query” for the input. Therefore, different from RNNs, the transformer model for a given input allows the next output token to be generated by determining which tokens in the sequence are important context for the given input and modeling the attention outputs as a summation of these tokens weighted by their importance. Ex. 2008, §1.2, §2.3.

54. Transformers offer several improvements over earlier technologies. One significant advantage is that transformers are able to fully utilize the parallel processing capabilities of GPUs and similar architectures like TPUs. Additionally, transformers overcome the prior limitations of relying only on nearby words to create the context for understanding words with multiple meanings. Transformers are a key factor in the success of systems like the popular ChatGPT, which are trained on vast amounts of data to perform tasks such as answering questions, text summarization, and translation. Pre-existing transformer models faced latency and throughput challenges, particularly in large-scale applications.

55. A representative transformer architecture is shown in FIG. B below. Given tokens for a user prompt, an attention layer during an “encoding phase” generates a key and a value for each input. The attention layer also generates a query for each input. For each query, the attention between the query and each of

the keys for the respective tokens in the sequence are computed. The attention output for the query is computed as a weighted sum of the attention and the corresponding value for each token.

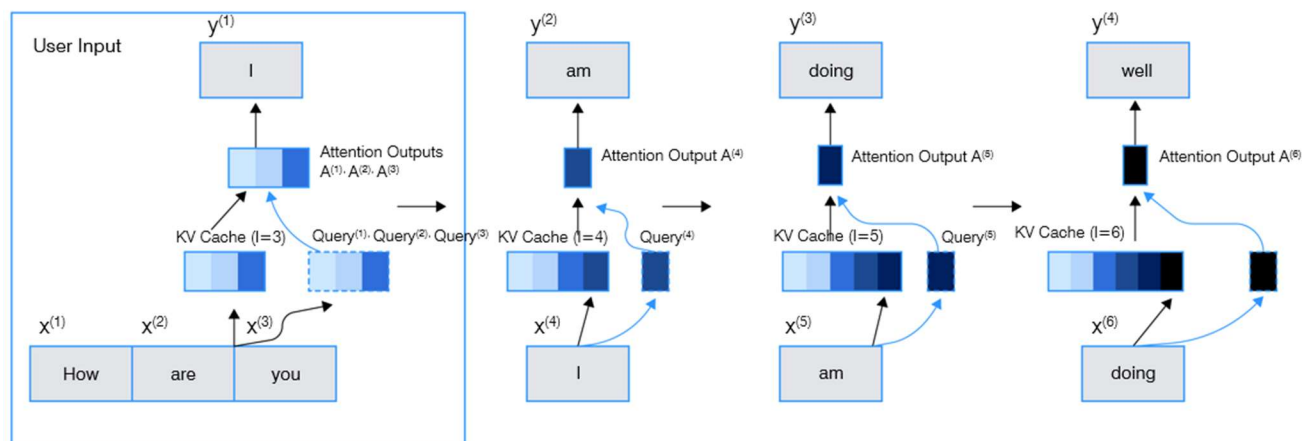


FIG. B. Overview of Transformer Model

56. For example, FIG. B illustrates tokens for a user prompt “How,” “are,” “you” that are initially input to an attention layer. The attention layer generates three keys and three values corresponding to each token. The attention layer also generates three queries corresponding to each token. The attention outputs are generated and subsequent operations on the attention outputs are used to generate a prediction for the next token “I.”

57. Due to the architecture of transformer models, the encoding phase of the transformer model typically can be performed in one forward pass iteration, whereas for the RNN model, the model is executed at least a number of iterations corresponding to the number of input tokens in the user prompt. The encoding phase of the transformer model therefore also reduces the number of iterations that

have to be executed as compared to RNNs. This difference is highlighted in the portions of FIGs. A and B that are marked “user input,” where multiple iterations are performed for the encoding phase of the example RNN, and one iteration is performed for the encoding phase of the example transformer model.

58. Moreover, different from the hidden state illustrated in FIG. A, transformer models service requests by maintaining a “KV Cache” for the attention layer that stores the “keys” (K) and the “values” (V) generated thus far for tokens in the sequence. Specifically, since the “keys” and “values” for a given token in the sequence will repeatedly be used to compute the attention outputs for subsequent iterations of inputs, the generated keys and values are stored in a “KV Cache” so that the keys and values can be retrieved for subsequent iterations without having to recompute them. Ex. 2007, §2.1.

59. The attention layer during one or more “decoding phases” iteratively generates output tokens until an end token <eos> (end-of-sequence) is generated. FIG. B illustrates at the next iteration, the previous output token “I” is used as the input and a key and a value for the input is generated and added to the KV Cache. The attention layer also generates a query for the input. The keys and values for the tokens are retrieved from the KV Cache. For the query, the attention between the query and each of the keys for the respective tokens in the KV Cache are computed. The attention output for the query is computed as a weighted sum of

the attention and the corresponding value for each token in the KV Cache. The subsequent operations are applied to the attention output to generate the next token “am.” This process is repeated for tokens “doing,” and “well,” and at each iteration, the keys and the values are generated for the input are added to the KV Cache. Therefore, different from the hidden states of RNNs, the number of elements in the KV Cache for an attention layer proportionally grows with respect to the length of the sequence.

60. Thus, a transformer model has an inherently different architecture from RNNs with respect to how sequential data is modeled through the attention layer. Specifically, as “keys” and “values” are generated for each token in a sequence, the relative importance of each individual token can be assessed, rather than aggregating the context of the entire sequence in a hidden state of fixed length in RNNs.

61. This difference results in many technical advantages that address the drawbacks of RNNs. Since the attention mechanism enables the transformer model to capture relationships between tokens regardless of their distance in the sequence, the model is adept at handling long-range dependencies. Ex. 2008, §2.3.2. This advantage is especially beneficial for complex tasks where context and relationships between distance elements are crucial for accurate interpretation and response. Ex. 2008, §2.3.2. Moreover, unlike RNNs that process sequential

data step-by-step, transformer models can process tokens in parallel (e.g., such as processing input tokens in the encoding phase), leading to faster training times.

Ex. 2008, §2.3.2.

C. Batching in RNNs vs. Transformer Models

62. Servers (e.g., cloud services) often handle requests from a large number of devices and employ “batching” to increase parallelism and throughput.

Ex. 2003, §1. Specifically, when requests are processed separately, a significant amount of energy consumption and communication cost is used to repeatedly access the learned parameter matrices of the trained model onto memory. Thus, during inference, batching merges multiple requests into a single batch so that the model processes the requests together. This reduces the number of accesses for parameter matrices and allows the energy cost to be shared among the requests in the batch. Ex. 2003, §1.

63. The operations in an RNN model that are batched requires lengths of the sequences to be the same across the batch of requests, i.e., the number of tokens in the user prompt and the number of iterations executed to be the same across different requests. Ex. 2003, §1. This is because during a batch operation, the data for the batch of requests are concatenated into a tensor, and the number of elements for a respective request in a tensor have to be the same across the requests in order for matrix multiplication to be performed efficiently. However, same-

length sequences are rarely the case in text modeling, as most requests have different user prompt lengths and different number of iterations until completion, both of which lead to variable sequence lengths. Ex. 2003, §1. Several methods have been proposed to address these difficulties of RNN batching.

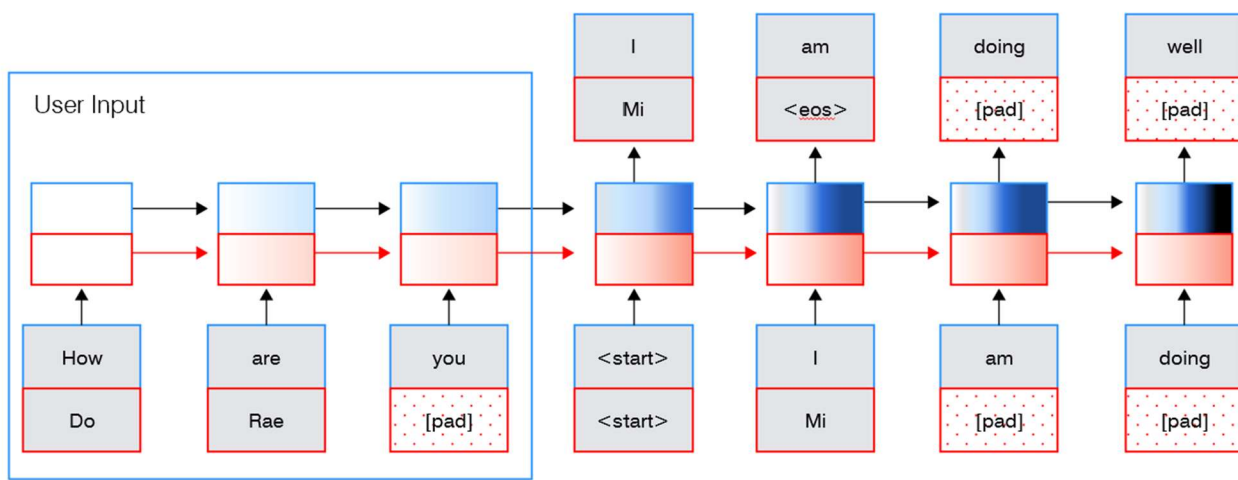


FIG. C. Batching in RNNs

64. One way was to perform “padding,” where padded tokens (e.g., zero tokens) are added to input tokens for the requests in a batch to match the length of the longest sequence in the batch. Ex. 2003, §1. FIG. C above illustrates an example of batching in RNNs for two requests. In addition to the request in FIG. A, the example in FIG. C adds an additional second request with tokens for user prompt “Do,” “Rae.” During the encoding phase, since the second request only has two input tokens compared to three input tokens for the first request, a dummy token is inserted for the second request at the third iteration.

65. During the decoding phase, after generating a first output token “Mi,” the second request is completed at the second decoding iteration as an end token $\langle \text{eos} \rangle$ is generated. However, since the second request is batched together with the first request, the inputs and output tokens for the second request continue to be padded with dummy tokens until the first request is completed. Therefore, a batch of requests start and complete together.

66. However, one main drawback of this batching is that the latency of all batched requests increases since the execution of requests cannot be completed until the longest sequence has been executed. Ex. 2003, §1. This results in underutilization of available parallel resources while waiting for the longest sequence to complete. Ex. 2003, §1.

67. Another way to perform batching for RNNs is “cellular batching,” which focuses on batching requests at the granularity of cells or some iterations, instead of whole sequences. Ex. 2003, §2.2.3. Unlike padding, new requests are allowed to join a batch whose execution is ongoing. Ex. 2003, §2.2.3. Also, once a request is completed, the response can be returned to the user. Ex. 2003, §2.2.3.

68. One reason why cellular batching is fairly feasible for RNNs is because the inputs to the RNN for the batch is typically a length of one token for each request, and the hidden state $h^{(t)}$ of the RNN model is of a fixed length regardless of the number of iterations. Therefore, batch operations that involve the

hidden states could be performed by concatenating the hidden states of the batch of requests together regardless of whether the requests of the batch were executed at a different iteration number.

69. Although transformer models were superior models in many aspects compared to RNNs, batching in transformer models remained a challenging problem. This is because for a batch of requests, the KV Cache for each request had to have the same lengths in order for batching to be performed, i.e., both the length of the user prompt tokens for a batch of requests and the number of iterations until the batch of requests were completed had to be the same. However, this scenario rarely occurs since the KV Cache grows with the length of the sequence for transformer models, and different requests almost always generate a different number of output tokens until completion. While “padding” could be applied to transformer models to add padded tokens to the inputs and/or the KV Caches for the requests in a batch, it would result in similar latency and throughput drawbacks as described above.

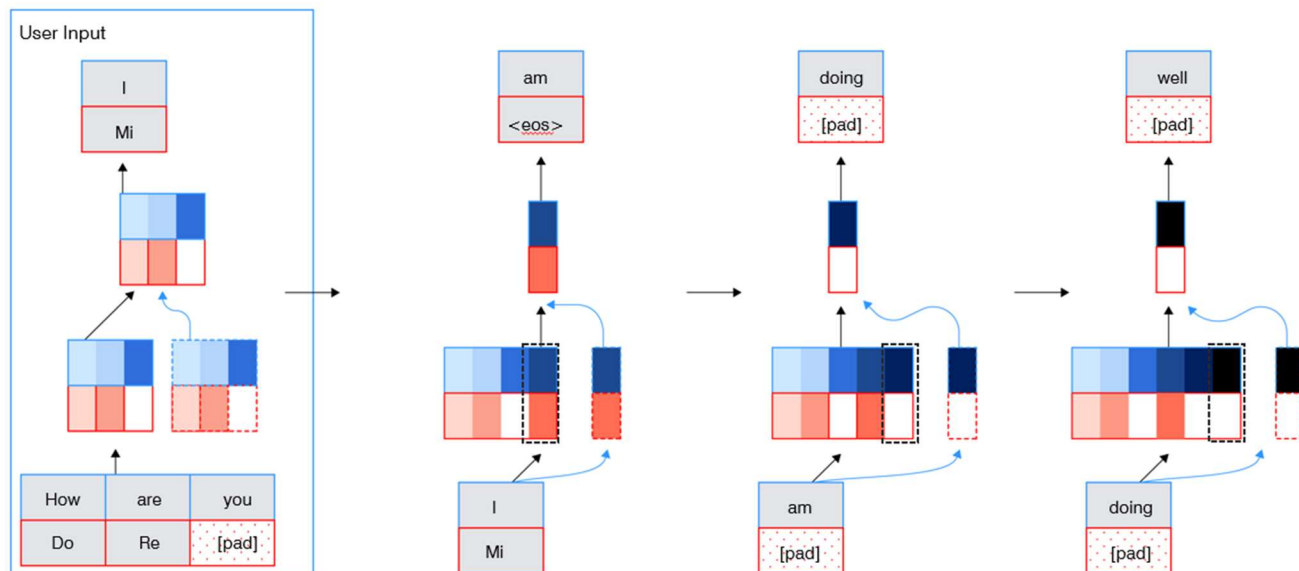


FIG. D. Static Batching in Transformer Models

70. FIG. D above illustrates static batching in transformer models.

During the encoding phase, the input tokens for the second request are padded with a one dummy token to match the length of three tokens for the first request. The first output tokens are generated for each of the requests. The batch of the first output tokens are input to the transformer model at the next iteration to generate second output tokens. The second request is completed since the end of sentence token <eos> is generated. However, since the requests are batched together, dummy tokens are input and generated as output tokens for the second request until the first request is completed.

D. Dynamic Batching in Transformer Models

71. The claimed invention solves the problem of conventional batching in transformers, as described above. FIG. E below illustrates dynamic batching in transformer models.

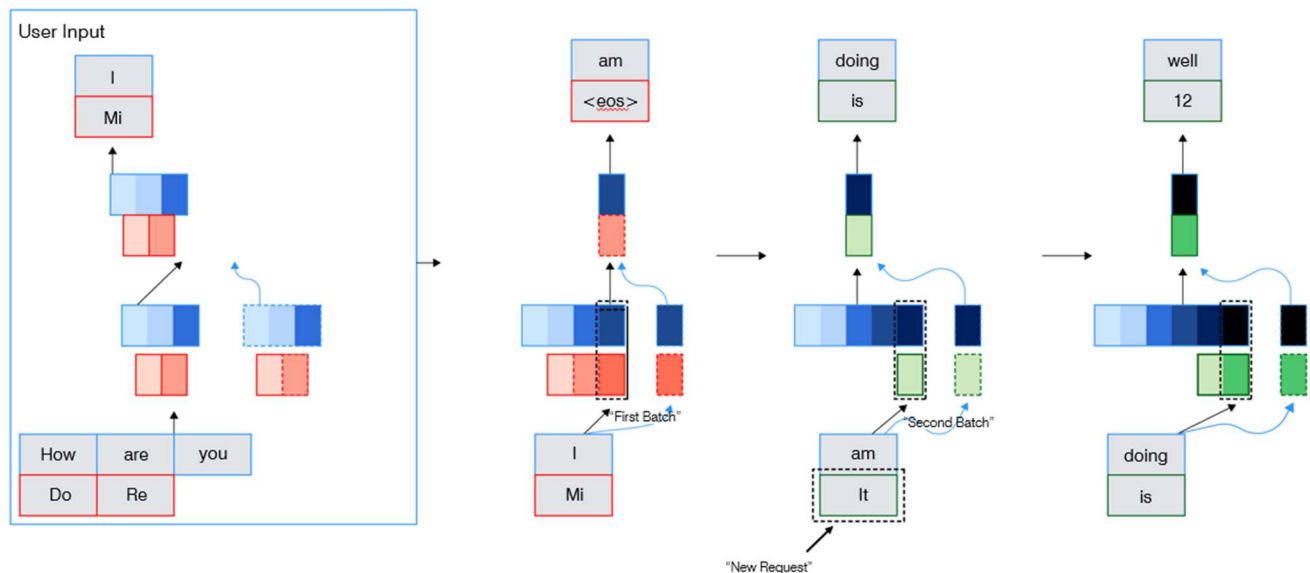


FIG. E. Dynamic Batching in Transformer Models

72. The batch of output tokens “I” for a first (blue outline) request and “Mi” for a second (red outline) request become a first set of inputs observed in the dynamic batching procedure after processing the user input (leftmost box). The transformer model is applied to the batch of requests thereafter to generate a set of output tokens, “am” for the first request (blue outline) and “<eos>” for the second request (red outline). The second request is completed since the end of sentence token <eos> is generated.

73. Since the batch can be modified in dynamic batching, the second request is removed and an input for a new request “It” is obtained. The batch of tokens “am” for the first request (blue outline) and “It” for the new request (green outline) become a new set of inputs for a second batch of requests that additionally include the new request. The transformer model is applied to the new batch of requests to generate the next set of output tokens, “doing” for the first request (blue outline) and “is” for the new request (green outline). This way, requests that are completed in a batch can be returned earlier to the user, and a new batch can additionally include a new request at an earlier point in time, reducing latency. Moreover, dummy tokens may not be needed since the length of the user prompt tokens or the length of the KV Cache are not required to be the same.

VII. OVERVIEW OF THE PRIOR ART

A. Katharopoulos

74. The Katharopoulos reference is titled “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention,” by Katharopoulos et al., published in 2020 (“Katharopoulos”; “Ex. 1005”). However, in my opinion the title of this article is misleading. Transformers *are not* RNNs. The Katharopoulos authors admit this. “In literature, transformer models are considered to be *a fundamentally different approach* to recurrent neural networks.” Ex. 1005, 5

(emphasis added); *see also* Ex. 2019 (“RNNs are not Transformers (Yet)”). Some of those differences are the ones I described above.

75. Transformers represent a revolutionary, generational advance in neural network design over their RNN predecessors, and serve as the basis for AI quantum leap of AI development and applications of the past few years, including use in ChatGPT. Katharopoulos states that transformers, since their introduction by Vaswani in 2017, “have demonstrated impressive results” on tasks such as writing natural language or deciphering audio speech. Ex. 1005, 1. A downside to transformers is they “often come with a very high computational and memory cost.” *Id.*, 1. The reason for these high computational and memory costs are attributed to “self-attention, which processes contexts of N inputs with a quadratic memory and time complexity $O(N^2)$.” Ex. 1005, 1.

76. Katharopoulos proposes a technique that he calls “linear transformers” that operate on a modified architecture. Ex. 1005, 2. These linear transformers perform *similarly to a standard transformer* but, it is claimed, are “up to 4000x faster on autoregressive prediction of very long sequences.” *Id.*, 1. The Katharopoulos linear transformer “significantly reduces the memory footprint and scales linearly with respect to the context length.” *Id.* Katharopoulos achieves this result by *rewriting a simplification of the transformer as an RNN*. *Id.*, 2 (“in §3.4 we rewrite the transformer as a recurrent neural network.”).

Katharopoulos asserts the transformer “can be written as a model that, given an input, modifies an internal state and then predicts an output, namely a Recurrent Neural Network (RNN).” Ex. 1005, 5 (§3.4).

77. In the standard transformer model, the attention layer enables the transformer to perform its complex reasoning tasks. Katharopoulos teaches “lineariz[ing] the attention computation.” Ex. 1005, 2. In other words, it handicaps the attention layer. It rewrites the transformer attention layer code from an “*exponential* kernel” into a simpler “linear” kernel. *Id.*, 3. The linear kernel then performs an attention calculation that generates *approximated* attention outputs. *Id.*, 2 (“Recent works ... have approximated softmax with a linear dot product... [W]e linearize the softmax attention in transformers”). Even Katharopoulos admits, “the feature function that corresponds to the exponential kernel is infinite dimensional, which makes *the linearization of exact softmax attention infeasible*.” *Id.*, 3 (§3.2) (emphasis added). The net effect is to improve computation speed but degrade output precision and quality. *Id.*, 2 (“we use the kernel formulation to speed up the calculation of self-attention”). Katharopoulos Table 3 shows reduced accuracy of linear attention accuracy as compared to the transformer in a speech recognition task. Ex. 1005, 8.

78. Since Katharopoulos essentially changes the attention mechanism in the transformer model, Katharopoulos does not maintain a KV Cache that stores the keys and values for individual tokens and grows in length as iterations are performed as described above. Neither does Katharopoulos compute the attention outputs in the way a transformer does.

79. Katharopoulos teaches equation 20 as one of the core mathematical manipulations that is necessary to rewrite a transformer into an RNN:

$$y_i = f_l \left(\frac{\phi(x_i W_Q)^T s_i}{\phi(x_i W_Q)^T z_i} + x_i \right). \quad (20)$$

where the two types of hidden states for the converted RNN are given by:

$$s_i = s_{i-1} + \phi(x_i W_K) (x_i W_V)^T, \quad (18)$$

$$z_i = z_{i-1} + \phi(x_i W_K), \quad (19)$$

Ex. 1005, 5.

80. However, Petitioner continues to describe the neural network after application of equation 20 as a “transformer model,” which it ***is not***. The petition mistakenly states, “Katharopoulos teaches that a first set of output tokens (e.g., “ y_i , the i -th output”) by applying ***the transformer model (e.g., equation 20)*** to a first set of inputs...” Pet., 49 (emphasis added). However, once Katharopoulos arrives at equation 20, all that remains is ***an RNN architecture***. Ex. 1005, 5 (“The resulting RNN has two hidden states...”). The transformer-based neural network has been ***replaced***.

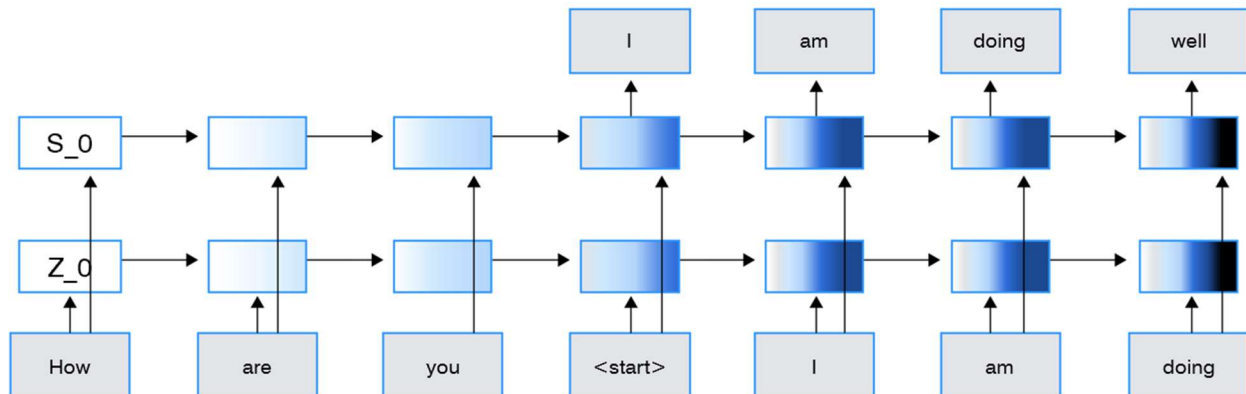


FIG. F. Example Representation of the Converted RNN in Katharopoulos

81. An example representation of the converted RNN in Katharopoulos is illustrated in FIG. F. As shown in the figure, the converted RNN is a variation of the RNN architecture shown in FIG. A that includes two hidden states, s_i and z_i . The two types of hidden states are generated over time and are used to generate the output tokens at each iteration.

82. Katharopoulos converts a transformer model to an RNN as illustrated in FIG. F above, with two hidden states s_i and z_i . While these two hidden states are computed using queries, keys, and values, it does not compute the attention outputs in the way that transformer models do, neither does the converted RNN maintain a KV Cache for doing so. Therefore, the converted RNN in Katharopoulos would suffer from the same difficulties as explained with respect to RNNs, and no longer retains the architecture of a transformer model.

In the first section of our supplementary material, we derive in detail the gradients for causally masked linear transformers and show that they can be computed in linear time and **constant memory**. In particular, we derive the gradients of a scalar loss with respect to the numerator of the following equation,

$$V'_i = \frac{\phi(Q_i)^T \sum_{j=1}^i \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^i \phi(K_j)}. \quad (21)$$

Ex. 1005, 11.

83. “In particular, the memory is constant with respect to the sequence length.” Ex. 1005, 7. “When it comes to inference [using the linear transformer], the cost per time and memory for one prediction is constant.” Ex. 1005, 4.

Furthermore, Katharopoulos shares the limitations of other RNN models such as the inability to parallelize training.

84. Others have also pointed out the shortcomings of Katharopoulos, including a recent paper titled “Transformers are not RNNs (yet).” *See* Ex. 2018. In another paper, “Recurrent Linear Transformers,” the author noted that the approach from Katharopoulos:

has three main limitations: (1) its self-attention mechanism naively adds positive values to the recurrent state, which can lead to instability when processing long sequences due to continual growth. (2) Performance is dependent on the choice of the kernel function—the element-wise feature maps used in the original paper, for example, have been shown to have limited memory capacity (Schlag et al., 2021). Lastly, (3) the Linear Transformer’s self-attention mechanism

maintains a matrix as a recurrent state, which can result in a high memory cost when multiple self-attention heads are used..

Ex. 2019, 1; *see also* Exs. 2015, 2016.

B. Gao

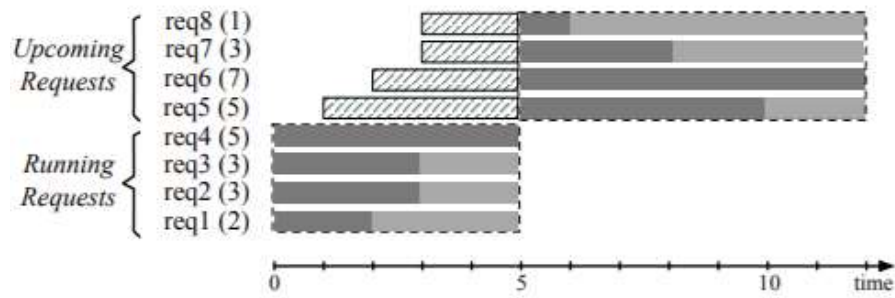
85. The Gao reference is titled “Low Latency RNN Inference with Cellular Batching” by Gao et al., published in 2018. Ex. 1004, 1. Gao proposes a technique called “cellular batching” to improve the latency and throughput of Recurrent Neural Network (RNN) inference. *Id.*

86. In layman’s terms, batching is the process of organizing a large pile of data into chunks of a predetermined size before feeding it into a neural network. *See* Pet., 19-20. Different batching techniques are commonly used to take advantage of different neural network architectures and hardware platforms. Pet., 21. When data inputs are uniform in length (i.e., they have the same “computation graph”), batching is a straightforward process. Ex. 1004, 3. This is the case for batching in the Multi-layer Perceptron (MLP) model or in the Convolution Neural Network (CNN) model. *Id.*

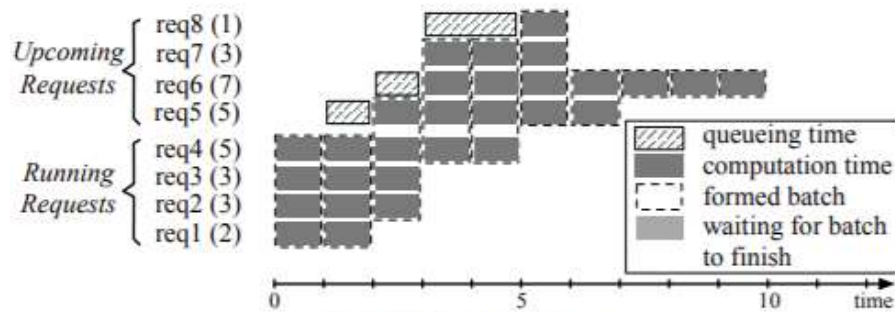
87. Common RNN batching techniques, referred to as “graph batching,” are inefficient. Ex. 1004, 4 (“Graph batching incurs extra latency due to unnecessary synchronization because an input cannot start executing unless all requests in the current batch have finished.”). While the RNN architecture “is

designed to model *variable length inputs*,” graph batching techniques work best with “inputs [of] *the same length*.” *Id.*, 1 (emphasis added). Thus, in practice, graph batching in RNNs can incur delays, and these delays are exacerbated “when inputs have varying lengths, causing some long input to delay the completion of the entire batch.” *Id.*, 4. Each variable-length input can have a different “recursion depth.” *Id.*, 3. As a result, the application of graph batching to RNNs presents a practical problem that “negatively affects both the latency and throughput.” *Id.*, 4.

88. Gao addresses this problem with a solution called “cellular batching.” Ex. 1004, 4. Gao’s key insight is to segment the recursive RNN data flow into many equally-sized cells, or “computation units,” which can be drawn as a sequence of cells in an organism. *Id.*, 2. Thus, Gao enables an RNN to perform batching at a smaller “cell” granularity instead of a larger “graph” granularity. *Id.* Gao illustrates this concept in Figure 5:



(a) Graph Batching



(b) Cellular Batching

Id., 4.

89. As Gao admits, the cellular batching solution is RNN-specific, and “cellular batching *is only beneficial for RNN inference.*” *Id.*, 14 (emphasis added). Although the Gao reference was published in 2018 and transformers were invented in 2017, Gao never suggests applying cellular batching to transformers. Gao never once mentions transformer models.

VIII. PETITIONER FAILED TO SHOW IT IS MORE LIKELY THAN NOT THAT THE CLAIMS ARE INVALID UNDER 35 U.S.C. §103

A. Petitioner Has Not Met Its Statutory Burden

1. The Petition Improperly Relies On Non-analogous Art

90. Gao and Katharopoulos are not analogous to the claimed invention.

The invention of the '775 patent “relates generally to machine-learning *transformer* neural network models, and more particularly to selective batching *for transformer models*.” Ex. 1001, 1:7-9 (emphasis added). The claims specifically recite methods and computer program products executing batches of requests using “a machine-learning *transformer* model.” See Ex. 1001, 27:55-30:47 (emphasis added). The specification distinguishes transformer models from “other recurrent machine-learning models (e.g., RNNs)” that use state data that have the same lengths across a batch of requests. Ex. 1001, 24:28-38. Neither Gao nor Katharopoulos pertain to batching for transformer models.

91. Neither Gao nor Katharopoulos are within this field of endeavor. Gao’s field of endeavor is batching on RNNs, rather than transformer models. Ex. 1004, 1 (Abstract), 14 (Conclusion). Katharopoulos’s field of endeavor is converting transformers into RNNs. Ex. 1005, 2. None of the disclosure of Gao or Katharopoulos is analogous to the embodiments, function, and structure of the claimed invention under the field of endeavor test.

92. The purpose of the '775 patent is to solve the problem of how to *process a batch of requests with variable lengths in transformer models* without using more resources. Ex. 1001, 1:48-51 (emphasis added); *see also id.*, Abstract. Gao's purpose is to improve both the latency and throughput of RNN inference, via cellular batching. Ex. 1004, 1 (Abstract). This is a different purpose than that of the '775 patent. Katharopoulos's purpose is to speed up autoregressive inference of certain transformers by "rewrit[ing] the transformer as a recurrent neural network." Ex. 1005, 2; *see also id.*, 1. Thus Katharopoulos also is directed to a different purpose than the '775 patent.

B. Defects in Obviousness Analysis

1. The Proposed Combination Suffers from Fatal Flaws for Failure to Show Certain Claim Elements

93. The proposed combination fails because it does not disclose or suggest several claimed limitations.

- a. **The proposed combination does not meet the claim limitations that explicitly recite "running," "access[ing]," and "applying" a transformer model (1[pre], 1[a], 1[c], 1[f]; 10[pre], 10[a], 10[c], 10[f])**

94. Claim 1 recites use of a transformer model throughout:

1[pre] A method of dynamically executing batches of requests on one or more execution engines **running a machine-learning transformer model**, comprising:

1[a] receiving, by a serving system, one or more requests for execution, the serving system including a scheduler and one or more execution engines each coupled to access a machine-learning transformer model including at least a set of decoders;

...

1[c] generating, by the execution engine, a first set of output tokens by applying the transformer model to a first set of inputs for the batch of requests, wherein applying the transformer model comprises applying at least one batch operation to one or more input tensors associated with the batch of requests;

...

1[f] generating, by the execution engine, a second set of output tokens by applying the transformer model to the second set of inputs for the second batch.

Ex. 1001, 27:55-28:20 (claim 1); *see also id.*, 29:1-36 (similar limitations of claim 10).³

95. As I discussed above, the very purpose of the '775 patent is to solve the problem of how to process a batch of requests with variable lengths *in transformer models* without using more resources. Ex. 1001, 1:48-51. Each of the

³ The arguments regarding claim 1 apply equally to claim 10. And although the dependent claims are not addressed herein, they inherit these deficiencies from the independent claims on which they depend.

four limitations above explicitly recite operations occurring in a transformer model. The transformer model is *not optional*.

96. Gao and Katharopoulos, alone or in combination, do not disclose or suggest batching *for transformer models*. Rather, the Petition’s proposed combination converts a transformer model to an RNN (Katharopoulos), and then applies “cellular batching” to the resulting RNNs (Gao). *See* Ex. 1005, 2; Ex. 1004, 1. As the Petition admits, Gao relates to RNNs and not a transformer model and that Katharopoulos “represents” the transformer model as an RNN. Pet., 40. Katharopoulos reformulates the “attention layer” of a transformer to convert the transformer model into an RNN. Ex. 1005, 5. At that point, *there is no longer a transformer model*. In the proposed combination, Gao is applied to the resultant RNN. Pet., 44-49. Therefore, if Katharopoulos was combined with Gao as the Petition proposes, the resultant combination at most would disclose iteration-level batching with RNNs, not transformers as claimed.

(1) **1[pre]: A method of dynamically executing batches of requests on one or more execution engines running a machine-learning transformer model, comprising**

97. Petitioner fails to show that the references render this limitation obvious. The Petition asserts that 1[pre] is disclosed by Gao, and argues that “a POSITA would have been motivated to combine Gao’s teaching of dynamically

executing batches of requests on one or more execution engines with

Katharopoulos's teaching of a running a machine-learning transformer model."

Pet., 38. The Petition does not explain how Gao's batches would be modified to execute on Katharopoulos's "transformer model." *Id.*

- (2) **1[a]: receiving, by a serving system, one or more requests for execution, the serving system including a scheduler and one or more execution engines each coupled to access a machine-learning transformer model including at least a set of decoders**

98. This limitation includes the requirement "one or more execution engines each coupled to access a machine-learning transformer model." The Petition relies on Gao's "workers" as the execution engines. Pet., 39. The Petition then discusses Katharopoulos's functionality, and then makes the conclusory statement that "Gao's Workers (i.e., execution engines) each have access to a machine-learning transformer model [of Katharopoulos] including at least a set of decoders." Pet., 41. Petitioner never says *how* Gao's workers would "each *[be]* *coupled to* access [the] machine-learning transformer model" of Katharopoulos, as required by the claims.

- (3) **1[c]: generating, by the execution engine, a first set of output tokens by applying the transformer model to a first set of inputs for the batch of requests, wherein applying the transformer model comprises applying at least one batch operation to one or more input tensors associated with the batch of requests**

99. The Petition relies on the combination as follows: “applying the transformer model (*e.g.*, equation 20) [of Katharopoulos] to a first set of inputs (*e.g.*, “ x_i denotes the i -th input”) for the batch of requests, *wherein applying the transformer model comprises applying at least one batch operation, as disclosed by Gao*, to one or more input tensors associated with the batch of requests.” Pet., 49 (emphasis added). This combination is problematic for the following reasons.

100. The limitation specifically recites applying *the transformer model* by *applying at least one batch operation*. As noted above, Gao’s batch operation is disclosed as being performed on RNNs, which is not “applying the transformer model,” as claimed. Moreover, once Katharopoulos arrives at equation 20, all that remains is an RNN architecture. Ex. 1005, 5. The transformer-based neural network has been replaced, so there is no “transformer model” present in formula 20.

- (4) **1[f]: generating, by the execution engine, a second set of output tokens by applying the transformer model to the second set of inputs for the second batch**

101. For this limitation, the Petition merely repeats its arguments from limitation 1[c]. Pet., 54-57. Thus, limitation 1[f] fails for the same reasons as limitation 1[c]. In addition, limitation 1[f] is not merely “a second iteration of the first ‘generating’ limitation” (1[c]). Petitioner relied on Katharopoulos for the “first set of inputs (*e.g.*, “ x_i denotes the i -th input”).” Pet., 49. However, for the “second set of inputs” in 1[e], the Petition relies upon a “second batch of requests” from Gao. Pet., 52. “[~~T~~he second set of inputs” of limitation 1[f] must be the same ones recited in limitation 1[e]. The Petition fails to explain how the “second batch of requests” from Gao could be *inputs* into formula 20 of Katharopoulos.

- b. **The combined references do not disclose or suggest “the second batch of requests scheduled responsive to determining that the execution engine has memory available to execute the second batch of requests” (1[e])**

102. Gao and Katharopoulos, alone or in combination, do not disclose or suggest this limitation. As I discussed above, transformer models maintain a KV Cache that stores the keys and values (*i.e.*, context) of tokens in the sequence, and which grows proportionally as the length of the sequence increases over time. Therefore, claim limitation 1[e] schedules the second batch of requests responsive to determining that the execution engine has memory available to execute the

second batch of requests. Since the amount of memory for storing the KV Cache for a request can grow over time and require more memory, this claim limitation first determines whether the execution engine has memory available to execute the second batch of requests before executing the second batch of requests.

103. In addition, neither Gao nor Katharopoulos teaches *determining whether an execution engine has memory available* for executing a second batch of requests. The proposed combination would not need such a memory check. The proposed combination at most teaches batching in RNNs, which have a fixed hidden state length across iterations. In particular, the “converted RNN” of Katharopoulos includes hidden states that are denoted as s_i and z_i , and that each have a fixed length across iterations of i . Ex. 1005, 5 (§3.4).

2. Petitioner Fails to Prove a Motivation to Combine

104. My understanding from counsel is that Petitioner has the burden to explain what changes are necessary to the references and *how* the substitution would meet the claim limitations. In my opinion, it has not done so.

a. A POSITA would not select and combine these references as proposed.

105. A POSITA *would not* select and combine these references when trying to solve the problem of the '775 patent. The combination would result in a system that makes more errors and takes longer to train than the transformers described in the '775 patent. It is the astonishing accuracy of systems built with

transformer that have led to their adoption in many AI applications over the earlier RNN systems.

106. Each of Petitioner’s proffered rationales for combining the references start with the assumption that a POSITA would have selected and combined these references. For example, Petitioner argues that a POSITA would have been motivated to combine the references because “[i]t is straight forward” to do so, and one reference’s technology “can be leveraged” to be used in the other. Pet., 27.

This does not show a POSITA would select and combine these references.

(1) **A POSITA faced with the problem of processing a batch of requests with variable lengths in transformers would not select the Gao and Katharopoulos references**

107. In particular, a POSITA would not have selected these references when faced with the problem the inventors sought to solve. The ’775 patent states that it sought to solve the problem of the potential infeasibility of processing a batch of requests with variable lengths in transformers. Ex. 1001, 1:46-51. As I discussed above, Gao and Katharopoulos are both from different fields of endeavor *and* serve a different purpose. A POSITA would not select—or even consider—references directed instead to simplifying transformers to allow rewriting them transformers as RNNs (Katharopoulos) and performing cellular batching in RNNs (Gao). Doing so in order to parallelize inference would prevent parallelization of

training, one key advantage of transformers over the older RNNs. A POSITA would not consider these references reasonably pertinent to the problem to be solved and Petitioner provided no rationale whatsoever to the contrary.

(2) **A POSITA would not have combined references in a way that eliminates the transformer model entirely**

108. The proposed combination writes out use of a transformer model entirely. A POSITA would not do this. Katharopoulos takes transformers and “approximates” them as RNNs. Specifically, Katharopoulos describes modifying the transformer layer to rewrite it as RNN. Ex. 1005, 5. Katharopoulos acknowledges the transformer is replaced at equation 20 shown below: “In the following equations, we formalize the transformer layer of equation 1 as a recurrent neural network. ***The resulting RNN*** has two hidden states...”

$$y_i = f_l \left(\frac{\phi(x_i W_Q)^T s_i}{\phi(x_i W_Q)^T z_i} + x_i \right). \quad (20)$$

Ex. 1005, 5.

109. Once that rewriting occurs, there is no longer any transformer model. The combination then applies cellular batching to the resultant RNN, as described in Gao. *See* Ex. 1004, 4. A POSITA would not, in my opinion, entirely rewrite one neural network model (transformer) to result in a different model (RNN).

(3) **A POSITA would not be motivated to modify a state-of-the-art transformer model to approximate a less sophisticated RNN model**

110. No POSITA would choose to go from a transformer model to an RNN model. To do so is to go backwards in time in the development and sophistication of machine learning models. RNNs were invented in the 1980's. Meanwhile, transformers have revolutionized AI research since their invention in 2017. *See* Ex. 1009, 1. As I discussed above, transformers were in fact developed to address some of the shortcomings of RNNs. For example, to avoid recursion in order to allow parallel computation, to eliminate vanishing and exploding gradients, and to reduce drops in performance due to long dependencies. Ex. 2008, §2.3.2. Transformer-based neural networks perform far better than older RNN models at complex tasks including pattern recognition, evaluating context, and making predictions. Ex. 2008, §1.2. Thus, to rewrite a transformer as an RNN model is essentially to handicap the transformer model, leading to an inferior result.

111. Moreover, as described above with respect to FIG. B, transformer models allow the encoding phase to be performed in one iteration, as the tokens for the user prompt can be processed together to generate the first output token, providing additional improvement in latency and computational efficiency. Since the proposed modification with Gao and Katharopoulos is a converted RNN, the proposed modification would require the encoding phase to be performed for a

number of iterations that correspond to the number of tokens in the user prompt, resulting in diminished latency compared to transformer models.

(4) **A POSITA would not be motivated to modify the references as proposed**

112. The proposed combination modifies the disclosure of the references to read in two features that are not taught or suggested by either reference. Rather, Petitioner fabricates these modifications out of whole cloth.

113. First, the references do not disclose or suggest mapping of the “Decoder layer” in Katharopoulos to “a single RNN cell.” Pet., 33. The Petition alleges that “[i]n the work of Katharopoulos, each Decoder layer is viewed as a single RNN cell (Decoder-Cell).” *Id.* It is not clear based on the Petition or the Expert Declaration what “a cell” of an RNN is or corresponds to. Even if there was a definition, Katharopoulos never mentions a “cell”; thus it is unclear how a POSITA would arrive at mapping a “Decoder layer” in Katharopoulos to a “RNN cell” of Gao. Petitioner never provides explains why a POSITA would be motivated to make this proposed modification. Indeed, Petitioner does ***not even acknowledge*** that the combination requires modification of the references.

114. Second, the references do not disclose or suggest generation of a (L x n) “computational graph.” Pet., 34-35. The Petition alleges that a (L x n) “computational graph” of RNN cells can be created based on the combination of

Katharopoulos and Gao that would enable cellular batching to be performed on batches of requests: “The computational graph . . . can be viewed as a matrix of Decoder-Cells with size (L x n). Each layer . . . comprises identical cells that process the different encoded input elements and only differ in the state-vector.”

Id. However, Katharopoulos appears to discuss the conversion of an attention mechanism to an RNN for only *a single* Decoder layer. It is unclear how a POSITA would arrive at a computational graph comprising multiple layers (or rows) of RNN cells from the teachings of the references. Again, Petitioner does *not even acknowledge* that the combination requires modification of the references, much less provide an explanation of why a POSITA would be motivated to make the proposed modification.

b. Petitioner provides no valid rationale for the combination

(1) Petitioner’s rationale neglects to consider the disadvantages of the combination

115. A POSITA would not consider the various features of references in a vacuum. Rather, a POSITA would consider both the advantages and disadvantages of a modifying or combining prior art references. Here, Petitioner points to the combination as “improv[ing] the latency and throughput” of the neural network and producing “significant speed ups.” Pet., 21, 31. Yet Petitioner is silent on the drawbacks of the combination.

116. RNNs have several disadvantages as compared to transformers, as I discussed above. For example, any “speed ups” made by the proposed combination would be offset by performance reductions in other areas, such as the neural network’s attention, context, and predictive ability. Ex. 2005, §2. RNNs perform much worse in terms of memory loss and at complex tasks including pattern recognition, evaluating context, and making predictions. *Id.* In addition, the RNN model that would result from the combination would need to be retrained, which is necessarily slow because RNNs require sequential processing, where each iteration during training depends on the output token of the previous iteration. *Id.*

117. Petitioner never acknowledges this need, never explains how this retraining would be accomplished on the resultant RNN, and never accounts for the added expenditure of time and resources. Indeed, Katharopoulos acknowledges that RNNs are less efficient to train than transformers. For example, for a given training sequence of 10,000 tokens, you only need to go through the transformer once (all tokens are simultaneously processed), but for RNNs, you have to iterate 10,000 times Ex. 1005, 4 (§3.3.2). Petitioner ignores the impact of these disadvantages of RNNs.

(2) **Petitioner relies on fictional, unsupported rationales**

118. I noticed that Petitioner’s motivation to combine analysis does not cite to Katharopoulos or Gao even once in the more than 10 pages it dedicates to this topic. *See* Pet., 27-37. For much of its analysis, it was unclear to me reading it whether Petitioner was relying on the disclosure of Gao, of Katharopoulos, or on neither reference, due to the lack of citations. *See, e.g.,* Pet., 32-33 (describing a “typical DOT circuit”).

119. In my opinion, Petitioner makes unexplained leaps of logic, leaving the reader to try to piece together the incomplete arguments. For example, Petitioner asserts that each transformer layer (“decoder block”) in the figure is mapped to a corresponding RNN cell. Pet., 30. However, Katharopoulos never once uses the word “cell” and Petitioner does not explain what in Katharopoulos it maps to a “cell” in Gao, and since Katharopoulos only describes one attention layer. A POISTA would not understand from this how to connect the other attention layers (1, 2, . . . , L) in a way that could be batched in Gao.

120. As another example, the Petition does not explain how Gao’s batching would apply to the linearized transformers of Katharopoulos that are its starting point *before* it rewrites them as RNNs. Petitioner relies on Katharopoulos as teaching that transformers can be rewritten as RNNs, and then batching in RNNs

as taught by Gao. Pet., 27-28. Petitioner then argues that the RNN batching of Gao can be modified so it applies to the transformers of Katharopoulos. Pet., 35-36. However, doing so would seem to require that Katharopoulos rewrite RNNs as transformers, which is the opposite of the rewriting that it teaches. Thus Petitioner's arguments appear to require logical leaps that simply do not follow from the disclosure of the references nor would be obvious to a POSITA.

121. Petitioner also relies on its lack of citations to misrepresent the record, which, in my opinion, is misleading at a minimum. For example, Petitioner states that Katharopoulos discloses that “even a softmax attention-function could be cast as an RNN cell” (Pet., 31) while making no citation to Katharopoulos to back up this statement. Yet Petitioner ignores that Katharopoulos explicitly states that “linearization of exact softmax attention [is] *infeasible*.” Ex. 1005, 3 (§3.2.1) (emphasis added).

(3) Petitioner's expert's parroted testimony cannot save its faulty analysis

122. As I discussed above, Petitioner's motivation to combine analysis contains zero cites to either reference. There does not appear to be *any* evidence, data, or other sources supporting Petitioner's rationale for combining the references. The Petition cites to 28 paragraphs of the declaration of Dr. Hammouri, but these paragraphs are a *nearly verbatim* parroting of the Petition

itself, differing by only a handful of words out of over 2,500, as I've shown below.

Pet., 27-37; Ex. 1002, ¶¶83-110.

IPR Petition , 27-37	Expert Report, ¶¶83-110
1. Summary of the Combination and Motivation to Combine It is straight forward to leverage the cellular batching presented by Gao to achieve an efficient scheduling scheme for at least certain classes of transformer models. Whereas the work of Gao shows the efficacy of cellular batching on a Recurrent Neural Network (RNN), Katharopoulos teaches a transformer model that operates like an RNN. As such, a POSITA understands the methodology of Gao can be leveraged to batch requests in the context of a machine learned transformer model. <i>See</i> Ex-1002, ¶83.	83. It is straight forward for a POSITA to leverage the paradigm of cellular batching presented by Gao to achieve an efficient scheduling scheme for at least certain classes of transformer models. Whereas the work of Gao shows the efficacy of cellular batching on a Recurrent Neural Network (RNN), Katharopoulos teaches a transformer model that operates like an RNN. As such, a POSITA understands the methodology of Gao can be leveraged to batch requests in the context of a machine learned transformer model.
This combination leverages two key observations that would be known to a POSITA: (1) whenever a Decoder-Only-Transformer (DOT) uses an attention mechanism with a linearized attention-function, it can be formulated as an RNN circuit as taught by Katharopoulos, and (2) transformers operate in a sequence-to-sequence fashion and thus, when DOTs are formulated as an RNN circuit they behave in a way that is similar to the Seq2Seq RNN model. <i>See generally</i> Ex-1018. <i>See</i> Ex-1002, ¶84.	84. This combination leverages two key observations that would be known to a POSITA: (1) whenever a Decoder-Only-Transformer (DOT) uses an attention mechanism with a linearized attention-function, it can be formulated as an RNN circuit as taught by Katharopoulos, and (2) transformers operate in a sequence-to-sequence fashion and thus, when DOTs are formulated as an RNN circuit they

IPR Petition , 27-37	Expert Report, ¶¶83-110
	behave in a way that is similar to the Seq2Seq RNN model. <i>See generally</i> Sutskever et al.
In view of the above, a POSITA would leverage Gao’s batching scheme to achieve efficient batching in a DOT with a linearized attention function as taught by Katharopoulos. <i>See</i> Ex-1002, ¶85.	85. With these two observations, a POSITA would be motivated to leverage Gao’s batching scheme to achieve efficient batching in a DOT with a linearized attention function as taught by Katharopoulos.
Gao’s Cellular Batching Gao disclose cellular batching that operates on the functional cell level of an RNN rather than on the whole data graph. In essence, the key insight of Gao is that, if batching is carried out on the cell level, then one can leverage the identical components in RNN cells in order to batch multiple requests and execute them all at the same time. Accordingly, Gao teaches batching any number of requests (depending on the underlying processing and memory capacity) and executing these batched requests in parallel. In addition, with cellular batching, requests of different sizes can be batched together without having to incur the cost of extra padding, another shortcoming of standard batching techniques. <i>See</i> Ex-1002, ¶86.	Gao’s Cellular Batching: 86. In their work, Gao presents a batching scheme (cellular batching) that operates on the functional cell level of an RNN rather than on the whole data graph as typically done in earlier graph batching schemes. In essence, the key insight of Gao is that: “RNN has the unique feature that it contains many identical computational units connected with each other. Cellular batching exploits this feature to 1) batch at the level of RNN cells instead of whole dataflow graphs, and 2) let new requests join the execution of current requests and let requests return to the user as soon as they finish.” Gao p. 4. This insight allows one to batch any number of requests (depending on the underlying processing and memory capacity) and to execute these batched

IPR Petition , 27-37	Expert Report, ¶¶83-110
	requests in parallel. In addition, with cellular batching, requests of different sizes can be batched together without having to incur the cost of extra padding, another shortcoming of standard batching techniques.
Finally, using cellular batching, newly arriving requests can join the execution of ongoing batches, while other requests can be returned to the user as soon as they finish, without the need to wait for the entire batch to finish simultaneously. <i>See</i> Ex-1002, ¶87.	87. Finally, using cellular batching, newly arriving requests can join the execution of ongoing batches, while other requests can be returned to the user as soon as they finish, without the need to wait for the entire batch to finish simultaneously.
Thus, Gao teaches cellular batching provides a significant improvement in terms of latency and throughput when running inference on requests of variable lengths. Gao built an inference system based on cellular batching that is called BatchMaker. <i>See</i> Ex-1002, ¶88.	88. Cellular batching has been shown to provide a significant improvement in terms of latency and throughput when running inference on requests of variable lengths. Gao built an inference system based on cellular batching that is called BatchMaker.
Katharopoulos Linearized Attention and Transformers as RNNs As discussed earlier, the attention mechanism has been used in many different works. In the work of Vaswani, the attention mechanism uses an attention function that computes the weight assigned to each Value using the exponential of the scaled dot product between Queries and Keys, which we refer to as a Softmax attention function. <i>See</i> Ex-1002, ¶89.	Katharopoulos Linearized Attention and Transformers as RNNs 89. As discussed earlier, the attention mechanism has been used in many different works. In the work of Vaswani, the attention mechanism uses an attention function that computes the weight assigned to each Value using the exponential of the scaled dot product between Queries and Keys, which we refer to as a Softmax attention function.

IPR Petition , 27-37	Expert Report, ¶¶83-110
Many papers have discussed specifics and features of different attention-functions (<i>see</i> Ex-1019 and Ex-1021); however, most relevant to our discussion here is the class of attention-functions introduced in Katharopoulos. <i>See</i> Ex-1002, ¶90.	90. Many papers have discussed specifics and features of different attention-functions (e.g., Tsai et al.); however, most relevant to our discussion here is the class of attention-functions introduced in Katharopoulos.
Katharopoulos discloses linearized attention-functions which are defined as a linear dot-product of kernel feature maps. In this type of attention-function, the attention mechanism itself still relies on the Query, Key and Value extracted in the same way defined in Vaswani. However, Katharopoulos that rather than using an exponential function directly on the product of the Query and the Key, a kernel feature map is first applied to the Query and Key values before their dot product is computed. <i>See</i> Ex-1002, ¶91.	91. In their work, Katharopoulos introduces linearized attention-functions which are defined as a linear dot-product of kernel feature maps. In this type of attention-function, the attention mechanism itself still relies on the Query, Key and Value extracted in the same way defined in Vaswani. However, rather than using an exponential function directly on the product of the Query and the Key, a kernel feature map is first applied to the Query and Key values before their dot product is computed.
To this end, Katharopoulos uses this class of attention-function to reduce the complexity of Attention computation and dramatically accelerate the autoregressive nature of Transformers especially during inference. Moreover, this type of attention-function can help in increasing the context length without losing on efficiency. In fact, Katharopoulos used linearized attention-functions to show that any	92. To this end, Katharopoulos uses this class of attention-function to reduce the complexity of Attention computation and dramatically accelerate the autoregressive nature of Transformers especially during inference. Moreover, this type of attention-function can help in increasing the context length without losing

IPR Petition , 27-37	Expert Report, ¶¶83-110
transformer layer that uses masked attention can be written as an RNN cell. <i>See</i> Ex-1002, ¶92.	on efficiency. In fact, Katharopoulos used linearized attention-functions to show that any transformer layer that uses masked attention can be written as an RNN cell.
More precisely, using a linearized attention-function, Katharopoulos shows that a Transformer's Decoder layer can be viewed as an RNN computational cell (Decoder-Cell) that takes in a single encoded input from a sequence of inputs, modifies an internal state accordingly, and then produces an output. <i>See</i> Ex-1002, ¶93.	93. More precisely, using a linearized attention-function, Katharopoulos shows that a Transformer's Decoder layer can be viewed as an RNN computational cell (Decoder-Cell) that takes in a single encoded input from a sequence of inputs, modifies an internal state accordingly, and then produces an output.
When processing a sequence of inputs, the cell's internal state is initialized to zero when processing the first element in the sequence, and for every consecutive sequence element, the internal state is initialized using the internal state of the previous cell. Here, all the cells processing the input sequence are identical, as is the case in RNNs, and each cell has an output that can be passed to a different Decoder-Cell corresponding to a higher layer or the Decoder stack or can be processed to predict the next element in the sequence. <i>See</i> Ex-1002, ¶94.	94. When processing a sequence of inputs, the cell's internal state is initialized to zero when processing the first element in the sequence, and for every consecutive sequence element, the internal state is initialized using the internal state of the previous cell. Here, all the cells processing the input sequence are identical, as is the case in RNNs, and each cell has an output that can be passed to a different Decoder-Cell corresponding to a higher layer of the Decoder stack or can be processed to predict the next element in the sequence.
Katharopoulos significantly improves the cost of computing attention (the most expensive component within a Transformer). For a context window	95. The work of Katharopoulos presents a radical improvement in the cost

IPR Petition , 27-37	Expert Report, ¶¶83-110
of length N, a softmax attention-function requires a time and memory complexity of $O(N^2)$ to compute, whereas using a linearized attention-function the same can be computed using a time and memory complexity of $O(N)$. <i>See</i> Ex-1002, ¶95.	of computing attention (the most expensive component within a Transformer). For a context window of length N, a softmax attention-function requires a time and memory complexity of $O(N^2)$ to compute, whereas using a linearized attentionfunction the same can be computed using a time and memory complexity of $O(N)$.
Moreover, Katharopoulos teaches even a softmax attention-function could be cast as an RNN cell, given that masked Attention is used. In fact, the softmax attention-function itself has been approximated using a linear dot product of feature maps (<i>see</i> Ex-1021 and Ex-1020) which is why Katharopoulos predicts that it might be possible to use, even a softmax attention-function, with their introduced work. <i>See</i> Ex-1002, ¶96.	96. Moreover, Katharopoulos argues that, in theory, even a softmax attention-function could be cast as an RNN cell, given that masked Attention is used “it becomes evident that any transformer layer with causal masking can be written as a model that, given an input, modifies an internal state and then predicts an output, namely a Recurrent Neural Network (RNN).” Katharopoulos p. 5. In fact, the softmax attention-function itself has been approximated using a linear dot product of feature maps (Blanc et al. and Rawat et al.) which is why Katharopoulos concludes that it could be possible to use, even a softmax attention-function, with their introduced work “approximating the RBF kernel with random Fourier features

IPR Petition , 27-37	Expert Report, ¶¶83-110
	could allow us to use models pretrained with softmax attention.” Katharopoulos p. 8.
Combining Gao’s Cellular Batching with Katharopoulos’s Transformer Given the significant speed ups presented by cellular batching and linearized attention-functions, it is natural to merge these two works into a fast-batching scheme for the well-known Decoder-Only-Transformer (“DOT”) architecture. Specifically, the link between these two works becomes clear when casting a Transformers’ Decoder into an RNN cell. In this view, the computational graph of a DOT becomes similar to that of a Seq2Seq RNN model only using multiple layers, and thus, cellular batching can be immediately leveraged to batch inputs into the same cell-types as detailed in Gao. <i>See</i> Ex-1002, ¶97.	Combining Gao’s Cellular Batching with Katharopoulos’s Transformer 97. Given the significant speed ups presented by cellular batching and linearized attention-functions, a POSITA can merge these two works into a fastbatching scheme for the well-known Decoder-Only-Transformer (“DOT”) architecture. Specifically, the link between these two works becomes clear when casting a Transformers’ Decoder into an RNN cell. In this view, the computational graph of a DOT becomes similar to that of a Seq2Seq RNN model, and thus, cellular batching can be immediately leveraged to batch inputs into the same cell-types as detailed in Gao.
A typical DOT circuit is made of a stack of L layers of Decoders. Each Decoder layer has different parameters (learned during training) and takes in a request in the form of a sequence of n (up to a maximum of N) input elements, each encoded as a vector of dimension d, and outputs a set of output elements with the same sequence length and vector	98. To elaborate, I first explain how a DOT operates. A typical DOT circuit is made of a stack of L layers of Decoders. Each Decoder layer has different parameters (learned during training) and takes in a request in the form of a sequence

IPR Petition , 27-37	Expert Report, ¶¶83-110
dimensionality. So, a Decoder layer maps a $(d \times n)$ matrix into a $(d \times n)$ matrix. Stacking L layers of Decoders does not impact the mapping space as it will simply iterate different functional mappings over the $(d \times n)$ space until the final Decoder layer outputs the final output of size $(d \times n)$. To this end, a Transformer's final output is expected to be a prediction of the next element in the sequence of inputs, just as is the case in Seq2Seq RNNs. Thus, the final output of the Decoder stack is passed through prediction-component (typically used in sequence transduction models like Seq2Seq RNNs) which is made of a linear projection to the possible output space followed by a softmax function (this is the output-softmax and is not to be confused with the attention-softmax discussed earlier) to produce a probability distribution over the possible output space. <i>See</i> Ex-1002, ¶98.	of n (up to a maximum of N) input elements, each encoded as a vector of dimension d, and outputs a set of output elements with the same sequence length and vector dimensionality. So, a Decoder layer maps a $(d \times n)$ matrix into a $(d \times n)$ matrix. Stacking L layers of Decoders does not impact the mapping space as it will simply iterate different functional mappings over the $(d \times n)$ space until the final Decoder layer outputs the final output of size $(d \times n)$. To this end, a Transformer's final output is expected to be a prediction of the next element in the sequence of inputs, just as is the case in Seq2Seq RNNs. Thus, the final output of the Decoder stack is passed through a prediction-component (typically used in sequence transduction models like Seq2Seq RNNs) which is made of a linear projection to the possible output space followed by a softmax function (this is the output-softmax and is not to be confused with the attention-softmax discussed earlier) to produce a probability distribution over the possible output space.

IPR Petition , 27-37	Expert Report, ¶¶83-110
Finally, a sampler is used to choose the predicted output element using the produced probability distribution. Once the output is produced it is added to the n input elements that will now be encoded as a $(d \times (n+1))$ matrix and passed through the Decoder stack in order to predict the $n+2$ elements in the sequence. <i>See</i> Ex-1002, ¶99.	99. Finally, a sampler is used to choose the predicted output element using the produced probability distribution. Once the output is produced it is added to the n input elements that will now be encoded as a $(d \times (n+1))$ matrix and passed through the Decoder stack in order to predict the $n+2$ elements in the sequence.
This process is repeated until the final output is a special output element (e.g., <eos>) is generated. At this point, the sampler stops sampling and can return the response to the input request in the form of the newly generated elements. This is, in essence, how a standard DOT circuit operates. <i>See</i> Ex-1002, ¶100.	100. This process is repeated until the final output generated is a special output element (e.g. <eos>). At this point, the sampler stops sampling and can return the response to the input request in the form of the newly generated elements. This is, in essence, how a standard DOT circuit operates.
In the work of Katharopoulos, each Decoder layer is viewed as a single RNN cell (Decoder-Cell). The Decoder-Cell only inputs a single element from the sequence of input elements, where the input elements are also encoded as a single vector of dimension d (one column from the matrix discussed in the previous paragraph). In response, the Decoder-Cell outputs a vector of the same size as the input and updates the state-vector reflecting the internal state of the Decoder-Cell at the time-step of the input. The output of a Decoder-Cell within the first layer is then passed to a different Decoder-Cell at	101. In the work of Katharopoulos, each Decoder layer is viewed as a single RNN cell (Decoder-Cell). The Decoder-Cell only inputs a single element from the sequence of input elements, where the input elements are also encoded as a single vector of dimension d (one column from the matrix discussed in the previous paragraphs). In response, the Decoder-Cell outputs a vector of the same size as the input and updates the state-vector reflecting the internal state of the Decoder-Cell at

IPR Petition , 27-37	Expert Report, ¶¶83-110
the next layer (e.g., layer 2) while the updated state-vector is used for processing the next input element within the input sequence. <i>See</i> Ex-1002, ¶101.	the time-step of the input. The output of a Decoder-Cell within the first layer is then passed to a different Decoder-Cell at the next layer (e.g. layer 2) while the updated state-vector is used for processing the next input element within the input sequence.
Accordingly, an input of n elements passing through a DOT with a linearized attention-function has a computational graph which can be viewed as a matrix of Decoder-Cells with size $(L \times n)$. Each layer (row in the computational graph) comprises identical cells that process the different encoded input elements and only differ in the state-vector passed from the previous (identical) Decoder-Cell within the same computational row. <i>See</i> Ex-1002, ¶102.	102. Accordingly, an input of n elements passing through a DOT with a linearized attention-function has a computational graph which can be viewed as a matrix of Decoder-Cells with size $(L \times n)$. Each layer (row in the computational graph) comprises identical cells that process the different encoded input elements and only differ in the state-vector passed from the previous (identical) Decoder-Cell within the same computational row.
The first Decoder-Cells (in all layers) are initialized with a zero state, as shown in Katharopoulos (Eq. 16 and 17). The final layer of Decoder-Cells do not pass their output anywhere and only update the state of the next Decoder-Cell in the same final layer (computational row). The only exception here is the final Decoder-Cell of the final layer (the top right Decoder-Cell of position (L,d) in the computational graph) which passes its output through the prediction-component (can be viewed as a	103. The first Decoder-Cells in all layers (the first column in the computational graph) is initialized with a zero state, as shown in Katharopoulos (Eq. 16 and 17). The final layer of Decoder-Cells does not pass their output anywhere and only update the state of the next Decoder-Cell in the same final layer (computational row). The only exception here is the final Decoder-Cell of the final

IPR Petition , 27-37	Expert Report, ¶¶83-110
different computational cell) made of the final linear projection and the output-softmax function in order to produce the predicted next token using a sampler as previously explained. <i>See</i> Ex-1002, ¶103.	layer (the top right Decoder-Cell of position (L,d) in the computational graph) which passes its output through the prediction-component (can be viewed as a different computational cell) made of the final linear projection and the output-softmax function in order to produce the predicted next token using a sampler as previously explained.
Just as done in Seq2Seq RNNs, the predicted output element is now passed through another column made of L Decoder-Cells identical to the previous Decoder-Cells used in each column of the computational graph. Moreover, the Decoder-Cells in the new column get their states initial value from the previous column of Decoder-Cells. That is, the computational graph can now be seen as being made of (L x (n+1)) decoder cells. The new Decoder-Cell column behaves exactly as the last Decoder-Cell column in that it passes its final output through a prediction-component cell to predict the next element. And just as done in the standard DOT architecture (or Seq2Seq RNNs), this process continues until the special element (e.g. <eos>) is generated, at which point the computational graph is terminated. <i>See</i> Ex-1002, ¶104.	104. Just as done in Seq2Seq RNNs, the predicted output element is now passed through another column made of L Decoder-Cells identical to the previous Decoder-Cells used in each column of the computational graph. Moreover, the Decoder-Cells in the new column get their states initial value from the previous column of Decoder-Cells. That is, the computational graph can now be seen as being made of (L x (n+1)) decoder cells. The new Decoder-Cell column behaves exactly as the last Decoder-Cell column in that it passes its final output through a prediction component cell to predict the next element. And just as done in the standard DOT architecture (or Seq2Seq RNNs), this process continues until the special element (e.g. <eos>) is generated, at which point the computational graph is terminated.

IPR Petition , 27-37	Expert Report, ¶¶83-110
As can be seen from the description here, this process is essentially identical to what is done in a Seq2Seq RNN model which also uses multi-levels of RNN cells and employs two types of cells, encoder-cells and decoder-cells. The only difference here is that the decoding process starts at the nth time-step rather than the $n+1$ time-step while only using a single type of cells. The rest is essentially identical in process to the Seq2Seq RNN model. The first n Cells in the first layer take in the encoded input elements (similar to encoder-cells in Seq2Seq), while for each newly generated element in the sequence the output element is encoded and passed as an input to the newly added Decoder-Cells (similar to the decoder cells in Seq2Seq). Thus, using the work of Katharopoulos, one can immediately see a DOT with linearized attention function as a Seq2Seq RNN circuit. <i>See</i> Ex-1002, ¶105.	105. As can be seen from the description here, this process is essentially identical to what is done in a Seq2Seq RNN model which can also use multi-levels of RNN cells and employs two types of cells, encoder-cells and decoder-cells. The only difference here is that the decoding process starts at the nth time-step rather than the $n+1$ time-step while only using a single type of cells. The rest is essentially identical in process to the Seq2Seq RNN model. The first n Cells in the first layer take in the encoded input elements (similar to encoder-cells in Seq2Seq), while for each newly generated element in the sequence the output element is encoded and passed as an input to the newly added Decoder-Cells (similar to the decoder cells in Seq2Seq). Thus, using the work of Katharopoulos, a POSITA can immediately appreciate a DOT with linearized attention function as a Seq2Seq RNN circuit.
Once casted in the language of a Seq2Seq RNNs, a DOT with linearized attention can immediately leverage cellular-batching exactly as presented in Gao. In essence, one can	106. Once casted in the language of RNNs, a POSITA would understand that

IPR Petition , 27-37	Expert Report, ¶¶83-110
use their BatchMaker system exactly as-is and without any changes to produce a batching system for DOT inference requests. The reason this application is immediate and straightforward is the ability to cast DOTs with linearized attention as an RNN with a computational graph that reuses the same cells for different steps of the computation. Any internal dependency between the Decoder-Cells, along with handling of variable-length requests, are directly handled by the BatchMaker system. See Ex-1002, ¶106.	a DOT with linearized attention can immediately leverage cellular-batching exactly as presented in Gao. In essence, one can use Gao’s BatchMaker system exactly as is and without any changes to produce a batching system for DOT inference requests. The reason this application is immediate and straightforward to a POSITA is the ability to cast DOTs with linearized attention as an RNN with a computational graph that reuses the same cells for different steps of the computation. Any internal dependency between the Decoder-Cells, along with handling of variable-length requests, are directly handled by the BatchMaker system.
A POSITA would fully appreciate the effect of applying BatchMaker to DOTs by observing the improvements that BatchMaker produces when applied to Seq2Seq RNNs. In this case, BatchMaker reports the ability to make interesting scheduling choices on multiple GPUs due to the presence of multiple Cell types. Moreover, in dynamic RNN models, the length of the computational path is not known beforehand, but rather is dependent on the output of the circuit itself, as is the case with Seq2Seq and DOTs. This feature presents a challenge to batching since even requests with identical input sizes might have an	107. A POSITA would fully appreciate the effect of applying BatchMaker to DOTs by observing the improvements that BatchMaker produces when applied to Seq2Seq RNNs. For example, in dynamic RNN models, the length of the computational path is not known beforehand, but rather is dependent on the output of the circuit itself, as is the case with Seq2Seq and DOTs. This feature presents a challenge to batching since even requests with identical input sizes might have an

IPR Petition , 27-37	Expert Report, ¶¶83-110
early or late finish depending on how the model responds to each request. In contrast, BatchMaker demonstrates how effective it is in these cases since cellular batching operates on the cell level which allows different requests to join or leave the execution at any time step. <i>See</i> Ex-1002, ¶107.	early or late finish depending on how the model responds to each request. In contrast, BatchMaker demonstrates how effective it is in these cases since cellular batching operates on the cell level which allows different requests to join or leave the execution at any time step.
Finally, applying Gao’s Cellular batching to other classes of Transformer circuits that are fundamentally different from RNNs might require further modifications. However, this is not the case for DOTs, which have a forward-moving dependency stemming from the use of Causal masking. With Causal masking, the direction of dependency is very similar to the forward passing state used in RNNs. Moreover, the attention mechanism, which is the defining characteristic of Transformers, requires input elements to, at least in the case of DOTs, have access to all the Keys and Values of all the previous input elements. <i>See</i> Ex-1002, ¶108.	108. Finally, it is worth noting here that a POSITA applying Gao’s Cellular batching to other classes of Transformer circuits, that are different from RNNs, may require further modifications. However, this is not the case for DOTs, which have a forward-moving dependency stemming from the use of Causal masking. With Causal masking, the direction of dependency is very similar to the forward passing state used in RNNs. Moreover, the attention mechanism, which is the defining characteristic of Transformers, requires input elements to, at least in the case of DOTs, have access to all the Keys and Values of all the previous input elements.
While in other transformer models the forward dependency must grow with the order of the input element (within a sequence of elements) in a way that requires an increasing number of computational cells to handle. However, this would only be the case	109. While in other transformer models the forward dependency must grow with the order of the input element (within a sequence of elements) in a way that

IPR Petition , 27-37	Expert Report, ¶¶83-110
for transformer models that use specific types of attention-functions that cannot linearize. <i>See</i> Ex-1002, ¶109.	requires an increasing number of computational cells to handle; however, this would only be the case for transformer models that use specific types of attention-functions that cannot linearize.
The attention mechanism relies on computing a weighted sum over the Values, with weights computed using a compatibility function between the Queries and the Keys. <i>See generally</i> Vaswani. Therefore, an intelligent choice of how compatibility is computed between the Queries and the Keys, as done in Katharopoulos, can easily allow the attention mechanism to be efficiently computed without the need for growth in the number of used computational cells. <i>See</i> Ex-1002, ¶110.	110. The attention mechanism relies on computing a weighted sum over the Values, with weights computed using a compatibility function between the Queries and the Keys. <i>See generally</i> Vaswani. Therefore, an intelligent choice of how compatibility is computed between the Queries and the Keys, as done in Katharopoulos, can easily allow the attention mechanism to be efficiently computed without the need for growth in the number of used computational cells.

123. Dr. Hammouri’s opinions about what a POSITA would “understand” from the references appear to be pure conjecture with no evidence to support them. I understand from counsel that such conclusory testimony cannot establish a motivation to combine and should be entitled to little weight.

(4) **Hindsight is the only possible rationale to combine Gao and Katharopoulos**

124. In my opinion, hindsight is the only rationale left. Faced with a lack of evidence to support its motivation to combine references that a POSITA would

not select or combine anyway, there is no proper basis for Petitioner’s dubious selection and combination of Gao and Katharopoulos. Petitioner’s attempt to Frankenstein together the prior art to result in the claims, based only on the teachings of the ’775 patent itself, should be rejected.

C. Dr. Hammouri’s Testimony Should Be Afforded Little Weight

125. Because Petitioner’s analysis is faulty and its expert’s testimony is conclusory, that testimony can’t save the Petition from its deficiencies. Dr. Hammouri’s expert testimony, in addition to being unsupported, consists of self-serving assertions, most of which parrot attorney argument verbatim from the petition. Similar to the portion I compared above, Pet., 38-45 and Ex. 1002, ¶¶120-134 are shown side by side below.

IPR Petition, 38-45	Expert Report, ¶¶120-134
1[PRE] A method of dynamically executing batches of requests on one or more execution engines running a machine-learning transformer model, comprising:	[PRE] A method of dynamically executing batches of requests on one or more execution engines running a machine-learning transformer model, comprising:
Gao teaches a system architecture of a “Manager and Worker. The manager processes arriving requests and submits batched computation tasks to workers for execution.” Ex-1004, 5. A color-coded annotation of Gao’s Figure 6 below shows how Gao’s system reads on	112. As discussed above, Gao proposes a system architecture of a “Manager and Worker. The manager processes arriving requests and submits batched computation tasks to workers for execution.” Gao at p. 5. A copy of the system architecture for BatchMaker is as follows:

IPR Petition, 38-45	Expert Report, ¶¶120-134
various claim limitations of the '775 Patent. <i>See</i> Ex- 1002, ¶120. Ex-1004, FIG. 6 (annotations added). Colorized annotations always added by Petitioner unless otherwise noted	
As shown in Figure 6, Gao includes a manager as a serving system that “processes arriving requests and submits batched computation tasks to workers for execution.” Ex-1004, 5. The manager in Gao has two components, a “request processor” for tracking the progress of execution for the requests, and a scheduler for “determining which cells from different requests would form a batched task, and selects a worker to execute the task.” Ex-1004, 5. <i>See</i> Ex-1002, ¶121.	As can be seen from Figure 6, BatchMaker includes a manager as a serving system that “processes arriving requests and submits batched computation tasks to workers for execution.” Gao at p. 5. The manager in BatchMaker has two components, a “request processor” for tracking the progress of execution for the requests, and a scheduler for “determining which cells from different requests would form a batched task, and selects a worker to execute the task.” Gao at p. 5.
Gao also includes an execution engine—namely workers—that use GPUs to “move[]a task from the task queue to the in-process queue once the task’s corresponding GPU kernel has been issued.” Ex-1004, 6. Using a pooling mechanism, the worker continues to move “finished tasks from the in-process queue and pushes it into the completion queue in the request processor.” Ex-1004, 6. Gao also discloses the BatchMaker system uses	BatchMaker also includes an execution engine, namely workers, that use GPUs to “move[]a task from the task queue to the in-process queue once the task’s corresponding GPU kernel has been issued.” Gao at p. 6. Using a pooling mechanism, the worker continues to move “finished tasks from the in-process queue and pushes it into the completion queue in the request processor.” Gao at p. 6.

IPR Petition, 38-45	Expert Report, ¶¶120-134
a set of “decoders” as well as noting “decoders” are common for popular RNN models. Ex-1004, 11. <i>See</i> Ex-1002, ¶122.	
As noted, Gao relates to RNN and not a machine-learning transformer model. However, a POSITA would understand Gao’s scheduling with dynamically batching cells would improve the latency associated with inference in using the machine-learning transformer model using a set of decoders, such as the linear transformer model taught by Katharopoulos. <i>See</i> Section (X)(B)(1) (motivation to combine discussion). <i>See</i> Ex-1002, ¶123.	As discussed above, Gao relates to RNN and not a machine-learning transformer model including at least a set of decoders. However, its dynamically batching cells would improve the latency associated with inference in a machine-learning transformer model such as Katharopoulos's linear transformer model which applies to a set of decoders since decoders are known to use causal masking. Motivation to Combine Discussion <i>Supra</i> .
Katharopoulos introduces a machine-learning transformer model that represents the transformer model as a Recurrent Neural Network (RNN). <i>See, e.g.</i> , Ex-1005, 5 (discussing that “from the causal masking formulation in 3.3 and the discussion in the previous section, it becomes evident that any transformer layer with causal masking can be written as a model that, given an input, modifies an internal state then predicts an output, namely a Recurrent Neural Network (RNN).”). <i>See</i> Ex- 1002, ¶124.	113. Katharopoulos introduces a machine-learning transformer model that represents the transformer model as a Recurrent Neural Network (RNN). <i>See e.g.</i> , Katharopoulos at p. 5 (discussing that “from the causal masking formulation in 3.3 and the discussion in the previous section, it becomes evident that any transformer layer with causal masking can be written as a model that, given an input, modifies an internal state then predicts an output, namely a Recurrent Neural Network (RNN).”).

IPR Petition, 38-45	Expert Report, ¶¶120-134
Katharopoulos formalizes the transformer layer for transformers using causal masking for decoder-only-transformer as an RNN in the following equation where x_i denotes the i-th input and y_i, the i-th output for a specific transformer layer: Ex-1004, 5. Importantly, Katharopoulos recognized that the “formulation does not impose any constrain on the feature function, and it can be used for representing any transformer model, in theory even the ones using softmax attention.” Ex-1005, 5. <i>See</i> Ex-1002, ¶¶125-126	114. Katharopoulos formalizes the transformer layer for transformers using causal masking for decoder-only-transformer as an RNN in the following equation where x_i denotes the i-th input and y_i, the i-th output for a specific transformer layer: 115. Importantly, Katharopoulos recognized that the “formulation does not impose any constrain on the feature function, and it can be used for representing <i>any transformer</i> model, in theory even the ones using softmax attention.” Katharopoulos at p. 5.
In view of the above, the combined teachings of Gao and Katharopoulos render obvious a system where Gao’s Workers (i.e., execution engines) each have access to a machine-learning transformer model including at least a set of decoders. <i>See</i> Section (X)(B)(1) (motivation to combine discussion). <i>See</i> Ex-1002, ¶127.	In view of the above, combining Gao and Katharopoulos would teach a system where Gao’s Batchmaker has access to a machine-learning transformer model including at least a set of decoders. <i>See also</i> Summary of the Combination and Motivation to Combine <i>Supra</i> (discussing the application of cellular batching to Decoder-Only-Transformer architectures).
1[B] scheduling, by the scheduler, a batch of requests including the one or more requests for execution on an execution engine;	1[B] scheduling, by the scheduler, a batch of requests including the one or more requests for execution on an execution engine;
As discussed above, Gao teaches a system architecture with a “scheduler.” Ex-1004, 5. As can be	116. As discussed above, BatchMaker proposes a system architecture of a “Manager and Worker. The manager

IPR Petition, 38-45	Expert Report, ¶¶120-134
seen from Figure 6, Gao includes a manager that “processes arriving requests and submits batched computation tasks to workers [i.e., execution engines] for execution.” Ex-1004, 5. The manager in Gao has two components, a “request processor” for tracking the progress of execution for the requests, and a scheduler for “determining which cells from different requests would form a batched task, and selects a worker to execute the task.” Ex-1004, 5. <i>See</i> Ex-1002, ¶¶128- 129. Gao provides that “[t]he scheduler needs to make decisions on what nodes should be batched together to form a task and what tasks to be pushed to which workers.” Ex-1004, 6. Gao goes on to provide detail for the aspects the scheduler must account for, namely “locality, priority, and utilization of multiple GPUs, which are often in conflict with each other.” Ex-1004, 6. “Locality refers to the preference that 1) the same set of requests should be batched together if they are to execute the same sequence of nodes, and the execution of the sequence of nodes should stick to the same GPU.” Ex-1004, 6. “Priority refers to the ability to prefer the execution of one type of cell over another.” Ex-1004, 6. <i>See</i> Ex-1002, ¶130.	processes arriving requests and submits batched computation tasks to workers for execution.” Gao at p. 5. A copy of the system architecture for BatchMaker is as follows:

IPR Petition, 38-45	Expert Report, ¶¶120-134
Gao also provides the basic algorithm employed by its BatchMaker system for scheduling and creation of batches. Ex-1004, 6. <i>See</i> Ex-1002, ¶131 Ex-1004, FIG. 6.	117. As can be seen from Figure 6, BatchMaker includes a manager that “processes arriving requests and submits batched computation tasks to workers for execution.” Gao at p. 5. The manager in BatchMaker has two components a “request processor” for tracking the progress of execution for the requests, and scheduler for “determining which cells from different requests would form a batched task, and selects a worker to execute the task.” Gao at p. 5. 118. Gao provides that “[t]he scheduler needs to make decisions on what nodes should be batched together to form a task and what tasks to be pushed to which workers.” Gao at p. 6. Gao goes on to provide detail for the aspects the scheduler must account for, namely “locality, priority, and utilization of multiple GPUs, which are often in conflict with each other.” Gao at 6. “Locality refers to the preference that 1) the same set of requests should be batched together if they are to execute the same sequence of nodes, and the execution of the sequence of nodes should stick to the same GPU.” Gao at p. 6. “Priority refers to the ability to prefer the execution of one type of cell over another.” Gao at p. 6.

IPR Petition, 38-45	Expert Report, ¶¶120-134
	119. Gao also provides the basic algorithm employed by BatchMaker for scheduling and creation of batches. Gao at p. 6.
1[C] generating, by the execution engine, a first set of output tokens by applying the transformer model to a first set of inputs for the batch of requests, wherein applying the transformer model comprises applying at least one batch operation to one or more input tensors associated with the batch of requests;	1[C] generating, by the execution engine, a first set of output tokens by applying the transformer model to a first set of inputs for the batch of requests, wherein applying the transformer model comprises applying at least one batch operation to one or more input tensors associated with the batch of requests;
As discussed above, Gao discloses an execution engine, namely workers, that use GPUs to “move[]a task from the task queue to the in-process queue once the task’s corresponding GPU kernel has been issued.” Ex-1004, 6. Using a pooling mechanism, the worker continues to move “finished tasks from the in-process queue and pushes it into the completion queue in the request processor.” Ex-1004, 6. In addition, Gao notes that “each request can return to the user as soon as its last cell finishes.” Ex-1004, 2. <i>See</i> Ex-1002, ¶132.	120. As discussed above, Gao discloses an execution engine, namely workers, that use GPUs to “move[]a task from the task queue to the in-process queue once the task’s corresponding GPU kernel has been issued.” Gao at p. 6. Using a pooling mechanism, the worker continues to move “finished tasks from the in- process queue and pushes it into the completion queue in the request processor.” Gao at p. 6; <i>see also</i> Gao at Fig. 6.

IPR Petition, 38-45	Expert Report, ¶¶120-134
Gao also batches requests, which is illustrated in Gao's cellular batching shown Figure 5 where requests 1-4 are initially batched together. <i>See</i> Ex-1002, ¶133. Ex-1004, FIG. 5	121. The behavior of Gao's cellular batching when being processed is illustrative Figure 5. Gao at p. 6; Gao at Fig. 5.
Gao describes this process as follows: In the beginning of time (t=0), the first 4 requests (req1-4) arrive... Under cellular batching, among the first 4 requests, the system forms two fully batched tasks, each performing the execution of a single (4- way batched) RNN cell. At t=2, the second task finishes, causing req1 to complete and leave the system. Since a new request (req5) has already arrived, the system forms its third fully batched task containing req2-5 at t=2. After finishing this task, another two existing requests (req2, req3) depart and two new ones are added (req6, req7) to form the fourth task. Ex-1004, 5.	122. Gao describes the process as follows: In the beginning of time (t=0), the first 4 requests (req1-4) arrive... Under cellular batching, among the first 4 requests, the system forms two fully batched tasks, each performing the execution of a single (4way batched) RNN cell. At t=2, the second task finishes, causing req1 to complete and leave the system. Since a new request (req5) has already arrived, the system forms its third fully batched task containing req2-5 at t=2. After finishing this task, another two existing requests (req2, req3) depart and two new ones are added (req6, req7) to form the fourth task. Gao at p. 6. (emphasis added).

126. Dr. Hammouri's expert testimony should be given little weight.

IX. CONCLUSION

I hereby declare that all statements made herein of my own knowledge are true and that all statements are made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: November 15, 2024

Michael Pazzani
Michael Pazzani, Ph.D.

Exhibit A

MICHAEL J. PAZZANI, Ph.D.

Career Summary

Recognized leader in artificial intelligence, machine learning and recommendation systems technology with accomplishments in academic, commercial and government R&D.

Department Head of Information and Computer Science leading growth in education and research programs.

Senior Research Official overseeing development of research collaborations, federal funding, corporate relations and technology commercialization.

Innovator who led a personalization company for mobile devices from initial concept to revenue.

Education

University of California, Los Angeles. Ph.D. Computer Science	1988
University of Connecticut. M.S. Computer Engineering	1981
University of Connecticut. B.S. Computer Engineering	1980

Experience

- Information Sciences Institute, University of Southern California, Los Angeles CA Research Scientist (2021-present)
- Halıcıoğlu Data Science Institute, University of California, San Diego, CA Distinguished Scientist (2019-2022)
- University of California, Riverside, Riverside, CA 2012-2019
Vice Chancellor for Research and Economic Development and Professor of Computer Science and Engineering, Statistics & Psychology (2012-2019)
 - Oversight of research and technology commercialization
 - Increased new federal grants from \$68M to \$131M
 - Conduct research on Deep Learning and Explainable Artificial Intelligence
- Rutgers, The State University of New Jersey, New Brunswick, NJ 2005-2012
Vice President of Research and Economic Development & Distinguished Professor of Computer Science (2006-2012)
 - Oversight of research and technology commercialization
 - Increased federal grants from \$129M to \$271M
 - Managed Technology Commercialization Office, increasing royalties by 80%
 - Conducted research on transfer learning and computational advertising
- National Science Foundation, Arlington, VA 2002-2006
Director, Information and Intelligent Systems
 - Managed budget of approximately \$250M to support basic research in artificial intelligence, data science, machine learning, robotics, internet search, computer vision, and human-computer interaction
 - Coordinated research program with Library of Congress and National Institutes of Health
- AdaptiveInfo, Irvine, CA 2000-2001
CEO and co-founder

- Innovated personalization methods for wireless web. Customers included Los Angeles Times, Sprint and San Diego Union Tribune. Sold to Versifi in 2001.
- University of California, Irvine, Irvine, CA 1988-2005
Chair, Information and Computer Science 1995-2000
Professor, Information and Computer Science 1988-2005
 - Conducted research on machine learning, personalization, internet search, recommendation systems, time series analysis, cognitive science, mobile devices and usability, medical informatics
 - Increased Corporate Support from \$0.6M to \$6M and federal funds from \$2M to \$6M
- Aerospace Corporation, El Segundo, CA 1984-1988
Member of the Technical Staff
 - Developed artificial intelligence systems to diagnose satellite anomalies
- Mitre Corporation, Bedford, M A 1980-1984
Member of the Technical Staff 1980-1983 & Artificial Intelligence, Group Leader 1983-1984
 - Developed artificial intelligence and natural language processing systems

Honors

AAAI Classic Paper Award recognized in 2014 for paper published in 1996 *“for significant contributions to the field of personalizing Internet content and learning user profiles”*.

Fellow of Association for the Advancement of Artificial Intelligence in 2003 *“for significant contributions to the development and understanding of machine learning algorithms and their relationship to human learning”*.

Board and Community Service

- Defense Science Board 2019-2021
Member: Provide the Secretary of Defense, with, independent advice and recommendations on scientific, technical, manufacturing, acquisition process, and other matters of special interest to the Department of Defense.
- Corporation for Education Network Initiatives in California (CENIC) 2013-2019
Member, Board of Directors: CENIC advances education and research statewide by providing a world-class network essential for innovation.
- Science, Technology, and Research Board, California Technology Council 2016-2019
Board Member. commercialization and to building scalable industry partnerships,
- Riverside ExCITE, Riverside, CA 2014-2016
Member, Board of Directors An incubator in partnership with business, government and UCR.
- Council on Competitiveness: Technology Leadership & Strategy Initiative 2014-2019
Member
- ACM International Conference on Intelligent User Interfaces, Palo Alto, CA 2011
Co-Chair
- New Jersey Research & Education Network (NJEDGE) 2010-2012
Member, Board of Directors
- NJEDGE provides collaborative resources and networked services in support of education and research.
- Commercialization Center for Innovative Technologies (CCIT) 2008-2012
Member, Board of Directors life science and technology incubator for NJ entrepreneurs.
- National Library of Medicine, Bethesda, MD 2003-2005
Board of Regents
- CombiChem, Inc, (NASDAQ: CCHM), San Diego, CA 1998-1999
Member, Board of Directors. CombiChem was a leader in computational drug design that was purchased by

FriendliAI Inc. / Hugging Face, Inc.

Exhibit 2001

Page 86

Most Important Publications (h-index 67- 40,000 citations)

Daniel Billsus, and Michael J. Pazzani (2007). Adaptive News Access, in "The Adaptive Web: Methods and Strategies of Web Personalization" (Springer, LNCS #4321), February 2007

Michael J. Pazzani and Daniel Billsus, (2007). Content-based Recommendation Systems, in "The Adaptive Web: Methods and Strategies of Web Personalization" (Springer, LNCS #4321), February, 2007

Hettich, S & Pazzani M (2006). Mining for Proposal Reviewers: Lessons Learned at the National Science Foundation. The Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining. Philadelphia.

Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra (2001). Locally adaptive dimensionality reduction for indexing large time series databases. SIGMOD 2001. Best paper award.

Bay, S.D., Pazzani, M.J. Detecting Group Differences: Mining Contrast Sets. Data Mining and Knowledge Discovery 5, 213–246 (2001). <https://doi.org/10.1023/A:1011429418057>

Pazzani, M. (2000). Knowledge discovery from data? IEEE Intelligent Systems 15(2): 10-13 (2000)

Pazzani, M. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review. 13(5-6): 393-408

Domingos, P. & Pazzani, M. (1997). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Machine Learning. 29, 103-130.

Pazzani M. & Billsus, D. (1997). Learning and Revising User Profiles: The identification of interesting web sites. Machine Learning 27, 313-331.

Pazzani, M., Muramatsu J. & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. Proceedings of the National Conference on Artificial Intelligence, Portland, OR.

Murphy, P. & Pazzani, M. (1994). Exploring the decision forest: An empirical investigation of OCCAM's razor in decision tree induction. Journal of Artificial Intelligence, 1, 257-275.

Pazzani, M. & Kibler, D. (1992). The role of prior knowledge in inductive learning. Machine Learning, 9, 54-97.

Pazzani, M. & Sarrett, W. (1992). A framework for average case analysis of conjunctive learning algorithms. Machine Learning, 9, 349-372.

Pazzani, M. (1991). The influence of prior knowledge on concept acquisition: Experimental and computational results. Journal of Experimental Psychology: Learning, Memory & Cognition, 17, 3, 416-432.

Pazzani, M. (1991). A computational theory of learning causal relationships. Cognitive Science, 15, 401-424.

Grants

Research Initiation: Induction of Causal Theories. National Science Foundation. July 15, 1989 - December 31, 1991. \$69,981.

Using Domain Knowledge to Influence Similarity Judgments. Science Applications International Corporation, McLean, VA 22102. April 1, 1990 - Sept 30, 1993. \$154,000.

Machine Learning Techniques for Modification and Maintenance of Expert System Knowledge Bases. NYNEX: 8/92 - 7/94. \$50,000.

Long and Medium-Term Research: Information-based Approaches to Learning Relational Concepts. National Science Foundation, 9/92 - 11/93. \$9550.

A Combined Analytic and Inductive Approach to Learning in Knowledge-Based Systems. AFOSR and ARPA, 9/30/92- 9/29/95. \$378,468.

Research in Machine Learning: Nippon Telephone and Telegraph. 6/93 - 7/94. \$22,500.

Minimizing misclassification costs: NYNEX 8/93 - 7/94. \$25,000.

Issues in Scaling up Machine Learning. AFOSR, 9/93 - 8/96. \$129,863

Learning Probabilistic Relational Concepts. National Science Foundation, 1994-1997. \$185,000.

Knowledge discovery in economic and political databases: Evidence Based Research, Arlington, VA 1994 - 1995. \$75,000.

Approximate Methods for Inference and Learning in Belief Networks. 1996 - 1997. AFOSR. \$138,454.

A Distributed Biomedical Computing Laboratory, National Science Foundation, 1997-1998. \$149,917 (PI: Lubimor Bic).

Graduate Fellowships in Biomedical Computing, Department of Education, 1997-2000. \$731,790 (PI: Lubimor Bic)

Knowledge Discovery in Problem Reports, TouchStone, 1997-1998. \$21,905.

From Computer Data to Human Knowledge: A Cognitive Approach to Knowledge Discovery and Data Mining, National Science Foundation, 1998-2001. \$240,000.

Graduate Assistance in Areas of National Need - UCI - ICS, Department of Education, 1998-2001. \$734,365.

An Online Repository of Large Data Sets for Data Mining Research and Experimentation, National Science Foundation, 1998-2000. \$99,737. (PI: Padhraic Smyth)

Adaptive Navigation in Information Space, Daimler Chrysler, 1998-1999. \$31,069.

Adaptive Navigation in Information Space, Daimler Chrysler, 1999-2000. \$26,000.

Organization and Navigation of Digital Documents, FileNET, 1999-2000. \$119,176

Intelligent Agents: Mitsubishi Electronics Corp. \$40,000 (1999)

Machine Learning: Hughes Research Laboratory. \$10,000 (1999)

Mobile Personalization, Qualcomm, equipment donation (1999).

Text Classification: Websense. 20,000 (2000).

E-commerce: The Irvine Company, \$10,000 (2000).

Transfer Learning, DARPA, \$225,000. 2006-2009

NJ-NANO: New Jersey Commission on Science and Technology: \$500K: 2006-2007

“Partnerships for Innovation” program to explore the use of agricultural materials in medical devices, food packaging, and fragrance delivery. NSF, \$600K (PI: K. Kohn): 2007-2009

Technology Commercialization Fund: New Jersey Commission on Science and Technology: \$500K: 2007-2008

NSF: Institutionalizing the IGERT Innovations at Rutgers, 2009-2014 \$1.25M (PI: P Furmanski)

Johnson & Johnson: Life Sciences Technology Commercialization \$350,000 (2008-10)

CC*IE Networking Infrastructure: University of California Riverside's Science DMZ. National Science Foundation. \$500,000. 2014-2015

Citrus Partnership. Eurosemillas, \$1.25M. 2015-2020.

Citrus Research Equipment at the University of California, Riverside, Economic Development Administration \$500,000 2017-18

Explainable Learning, DARPA via NAVAL RESEARCH LABORATORY \$720,000 2017-2021

Mobility Hub and Central Campus Neighborhood Improvements. Federal Transit Administration \$4.4M 2017-2020

Life Science Incubator for University of California, Riverside. Economic Development Administration \$2.5M 2019-2021

RAPID: Explainable Machine Learning for Analysis of COVID-19 Chest CT NATIONAL SCIENCE FOUNDATION, \$ 120,243.00, 2020-2021

Complete Publication List

Books

1. Pazzani, M. (1990). Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods. Hillsdale, NJ: Lawrence Erlbaum Associates.

Edited Books

1. Fisher, D., Pazzani, M. & Langley, P. (1991). Concept formation: Knowledge and experience in unsupervised learning. San Mateo, CA: Morgan Kaufmann.

Journal Articles

1. Cullingford, R. & Pazzani, M. (1984). Word-meaning selection in multiprocess language understanding programs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6,4, 493-509.
2. Pazzani, M. (1987). Explanation-based learning for knowledge-based systems. International Journal of Man-Machine Studies, 26, 413-433.
3. Pazzani, M. (1987). Failure-driven learning of fault diagnosis heuristics. IEEE Transactions on Systems, Man and Cybernetics: Special issue on Causal and Strategic Aspects of Diagnostic Reasoning, 17, 3, 380-394.
4. Pazzani, M. (1990). Learning in order to avoid search in logic programming. Computers and Mathematics with Applications, 2, 10, 101-110.
5. Pazzani M. & Dyer, M. (1990). Memory organization and explanation-based learning. International Journal of Expert Systems, 2, 3, 331-358.
6. Pazzani, M. (1991). A computational theory of learning causal relationships. Cognitive Science, 15, 401-424.
7. Pazzani, M. (1991). Learning to predict and explain: An integration of similarity-based, theory-driven and explanation-based learning. Journal of the Learning Sciences, 1, 2, 153-199.
8. Pazzani, M. (1991). The influence of prior knowledge on concept acquisition: Experimental and computational results. Journal of Experimental Psychology: Learning, Memory & Cognition, 17, 3, 416-432.
9. Pazzani, M. & Brunk, C. (1991). Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. Knowledge Acquisition, 3, 157-173.
10. Pazzani, M. & Kibler, D. (1992). The role of prior knowledge in inductive learning. Machine Learning, 9, 54-97.
11. Pazzani, M. & Sarrett, W. (1992). A framework for average case analysis of conjunctive learning algorithms. Machine Learning, 9, 349-372.
12. Pazzani, M. (1993). Learning causal patterns: Making a transition from data-driven to theory-driven learning. Machine Learning, 11, 173-194.

13. Murphy, P. & Pazzani, M. (1994). Exploring the decision forest: An empirical investigation of OCCAM's razor in decision tree induction. *Journal of Artificial Intelligence*, 1, 257-275.
14. Ali K. & Pazzani M. (1995). HYDRA-MM: Learning Multiple Descriptions to Improve Classification Accuracy. *International Journal on Artificial Intelligence Tools*, 4.
15. Ali K. & Pazzani M. (1996). Error Reduction through Learning Multiple Descriptions Machine Learning, 24:3.
16. Merz, C. Pazzani, M & Danyluk, A. (1996). Tuning Numeric Parameters to troubleshoot a telephone network. *IEEE Expert*, Feb 1996, pg. 44-49.
17. Shankle, W.R., Datta, P., Dillencourt, M. & Pazzani, M. (1996). Improving Dementia Screening Tests with Machine Learning Methods. *Alzheimer's Research*. 2(3).
18. Pazzani, M., See, D., Shroeder, E. & Tilles, J. (1997). Application of an Expert System in the Management of HIV-infected patients. *Journal of AIDS and Human Retrovirology*. 15:356-362.
19. Pazzani M. & Billsus, D. (1997). Learning and Revising User Profiles: The identification of interesting web sites. *Machine Learning* 27, 313-331.
20. Domingos, P. & Pazzani, M. (1997). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Machine Learning*. 29, 103-130.
21. Mani, S., Shankle, R., Dick, M & Pazzani, M. (1999). Two-Stage Machine Learning Model for Guideline Development. *Artificial Intelligence in Medicine*. 16, 51-80.
22. Merz, C. & Pazzani, M. (1999). A Principal Components Approach to Combining Regression Estimates Machine Learning. 36, 9-32.
23. Lathrop, R. & Pazzani, M. (1999). Combinatorial Optimization in Rapidly Mutating Drug-Resistant Viruses. *Journal of Combinatorial Optimization*. 3, 301-320.
24. Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S (2000) Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems* 3(3): 263-286.
25. Pazzani, M. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*. 13(5-6): 393-408
26. Pazzani, M. (2000). Learning with Globally Predictive Tests. *New Generation Computing* 18(1): 28-38
27. Pazzani, M. (2000). Knowledge discovery from data? *IEEE Intelligent Systems* 15(2): 10-13 (2000)
28. Billsus, D. & Pazzani, M. (2000). User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*. 10:2/3. 147-180
29. M. J. Pazzani, S. Mani, W. R. Shankle (2001). Acceptance of Rules Generated by Machine Learning among Medical Experts. *Methods of Information in Medicine*; 40: 380-385.

30. G. Webb, Michael J. Pazzani, Daniel Billsus, (2001).Machine learning for user modeling. User Modeling and User-Adapted Interaction 11: 19-20, 2001
31. Bay, S. D. & Pazzani, M. J. (2001). Detecting Group Differences: Mining Contrast Sets. Data Mining and Knowledge Discovery.
32. Daniel Billsus, Clifford A. Brunk, Craig Evans, Brian Gladish & Michael Pazzani (2002) Adaptive Interfaces for Ubiquitous Web Access, CACM p 34-38.
33. Pazzani, M. & Billsus, D. (2002). Adaptive Web Site Agents. Journal of Agents and Multiagent systems,5(2) 205-218
34. Keogh, E., Chakrabarti, K., Mehrotra S. & Pazzani, M (2002). Locally adaptive dimensionality reduction for indexing large time series databases. ACM Transactions on Database Systems, June 2002
35. Keogh, E. & Pazzani, M. (2002). Learning the Structure of Augmented Bayesian Classifiers. Vol. 11, No. 4 (2002) 587-601
36. Miyahara, K. & Pazzani, M. J. (2002). Improvement of Collaborative Filtering with the Simple Bayesian Classifier. IPSJ Journal, Vol.43, No.11, Information Processing Society of Japan, November, 2002
37. Rawassizadeh R; Tomitsch M; Nourizadeh M; Momeni, E; Peery, A Ulanova, L. & Pazzani, M (2015). Energy-Efficient Integration of Continuous Context Sensing and Prediction into Smartwatches. . *Sensors*, 22616-22645.
38. Rawassizadeh, R., Momeni, E., Dobbins, C., Gharibshah, J. Pazzani, M. (2016) Scalable Mining of Daily Behavioral Patterns from Multivariate Temporal Data. In IEEE Transactions on Knowledge and Data Engineering. 28(11), 3098-3112.
39. Rawassizadeh, R., Sen, T., Kim, S.J. Meurisch, C. Keshavarz,H. Mühlhäuser, M. & Pazzani, M.. CCF Trans. Pervasive Comp. Interact. (2019) 1: 163-174.
40. Rawassizadeh, R., Dobbins, C., Akbari, M., & Pazzani, M. (2019). Indexing Multivariate Mobile Data through Spatio-Temporal Event Detection and Clustering. *Sensors*,19(3), 448.
41. Rawassizadeh, R., Keshavarz, H., & Pazzani, M. (2019). Ghost Imputation: Accurately Reconstructing Missing Data of the Off Period. IEEE Transactions on Knowledge and Data Engineering.
42. Rawassizadeh, R., Sen, T., Kim, S. J., Meurisch, C., Keshavarz, H., Mühlhäuser, M., & Pazzani, M. (2019). Manifestation of virtual assistants and robots into daily life: Vision and challenges. CCF Transactions on Pervasive Computing and Interaction, 1(3), 163-174.
43. Ivaturi, P., Gadaleta, M., Pandey, A. C., Pazzani, M., Steinhubl, S. R., & Quer, G. (2020). Explaining the Deep Learning Black Box by Identifying Segments of the Single-Lead ECG Signal Used for Detection of Atrial Fibrillation. *Circulation*, 142(Suppl_3), A17189-A17189.
44. P. Ivaturi, M. Gadaleta, A. C. Pandey, M. Pazzani, S. R. Steinhubl and G. Quer, (2021) A Comprehensive Explanation Framework for Biomedical Time Series Classification, in IEEE Journal of Biomedical and Health Informatics, doi: 10.1109/JBHI.2021.3060997.

- 45 Huynh, J., Masoudi, S., Noorbakhsh, A., Mahmoodi, A., Kligerman, S., Yen, A., Jacobs, K., Hahn, L., Hasenstab, K., Pazzani, M. and Hsiao, A., 2022. Deep Learning Radiographic Assessment of Pulmonary Edema: Optimizing Clinical Performance, Training With Serum Biomarkers. *IEEE Access*, 10, pp.48577-48588.
- 46 Pazzani, M. Soltani, S., Kumar, K., Alipour, A and Ahamed, A. (2022). Improving Explanations of Image Classifiers: Ensembles and Multitask Learning. *International Journal of Artificial Intelligence and Applications (IJAIA)* 13.
- 47 Fan, R., Alipour, K., Bowd, C., Christopher, M., Brye, N., Proudfoot, J.A., Goldbaum, M.H., Belghith, A., Girkin, C.A., Fazio, M.A., Liebmann, J.M., Weinreb, R., Pazzani, M., Kriegman, D., and Zangwill, L. (2023). Detecting Glaucoma from Fundus Photographs Using Deep Learning without Convolutions: Transformer for Improved Generalization. *Ophthalmology Science*, 3(1), p.100233.
- 48 K. A. Hasenstab, J. Huynh, S. Masoudi, G. M. Cunha, M. Pazzani and A. Hsiao, "Feature Interpretation Using Generative Adversarial Networks (FIGAN): A Framework for Visualizing a CNN's Learned Features," in *IEEE Access*, vol. 11, pp. 5144-5160, 2023, doi: 10.1109/ACCESS.2023.3236575.

Book Chapters

1. Pazzani, M. (1988). Explanation-based learning for knowledge-based systems. In B. Gaines & J. Boose (Eds.), *Knowledge acquisition for knowledge-based systems*. (pp. 215-238). London: Academic Press.
2. Pazzani, M. (1989). Creating high-level knowledge structures from simple elements. In K. Morik (Ed.), *Knowledge representation and organization in machine learning. Lecture notes in Artificial Intelligence*, No 347. New York: Springer-Verlag.
3. Pazzani, M. & Flowers, M. (1990). Scientific discovery in the layperson. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation*. San Mateo, CA: MorganKaufmann.
4. Fisher, D. & Pazzani, M. (1991). Theory-guided concept formation. In D. Fisher, M. Pazzani & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.
5. Fisher, D. & Pazzani, M. (1991). Concept formation in context. In D. Fisher, M. Pazzani & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.
6. Pazzani, M. (1989). Learning fault diagnosis heuristics from device descriptions. In Y. Kodratoff & R. Michalski (Eds.), *Machine Learning: An artificial intelligence approach (Vol. III)*. San Mateo, CA: Morgan Kaufmann
7. Pazzani, M., Brunk, C. & Silverstein, G. (1992). A information-based approach to combining empirical and explanation-based learning. In S. Muggleton (Ed.). *Inductive Logic Programming* (pp 373-394), London: Academic Press.
8. Pazzani, M. (1994). Learning causal patterns: Making a transition from data-driven to theory-driven learning. In Ryszard Michalski & Georghe Tecuci (Eds.) *Machine Learning (Vol. IV): A Multi-Strategy Approach*. San Mateo, CA: Morgan Kaufmann. (Extended version of Journal paper 4.12).

9. Hirschberg, D., Pazzani, M. & Ali, K. (1995). Average case analysis of k-CNF and k-DNF learning algorithms. In S. Hanson, M. Kearns, T. Petsche and R. Rivest Computational Learning Theory and Natural Learning Systems: Constraints Prospects. Cambridge, MA: MIT Press.
10. Yamazaki, T., Pazzani, M. & Merz, C. (1996). Acquiring and updating hierarchical knowledge for machine translation based on a clustering technique. In Wermter, Riloff & Scheler (Eds.) Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing.
11. Pazzani, M. (1997). Searching for dependencies in Bayesian classifiers. Artificial Intelligence and Statistics IV, Lecture Notes in Statistics, Springer-Verlag: New York.
12. W. R. Shankle, Subramani Mani, Michael J. Pazzani & Padhraic Smyth. (1997). Detecting very early stages of dementia from normal aging with machine learning methods. In Keravnou, E., Garbay, C., Baud, R. & Wyatt, editors, Lecture Notes in Artificial Intelligence: Artificial Intelligence in Medicine, AIME97, volume 1211, pages 73-85, Springer.
13. Daniel Billsus, and Michael J. Pazzani (2007). Adaptive News Access , In "The Adaptive Web: Methods and Strategies of Web Personalization" (Springer, LNCS #4321), February 1, 2007
14. Michael J. Pazzani and Daniel Billsus, (2007).Content-based Recommendation Systems. In "The Adaptive Web: Methods and Strategies of Web Personalization" (Springer, LNCS #4321), February 1, 2007

Workshops

1. Pazzani, M. (1987). Inducing causal and social theories: a prerequisite for explanation-based learning. Proceedings of the Fourth International Workshop on Machine Learning (pp. 230-241). Irvine, CA: Morgan Kaufmann.
2. Pazzani, M. (1987). Creating high-level knowledge structures from primitive elements. Knowledge Representation and Knowledge Organization in Machine Learning Workshop. Geseke, Germany.
3. Pazzani, M. (1988). Selecting the best explanation for explanation-based learning. AAAI Symposium on Explanation-Based Learning (pp. 156-170). Stanford University.
4. Pazzani, M. (1988). Learning during plan recognition. AAAI Workshop on Plan Recognition. (pp. 1-5). St. Paul, MN.
5. Pazzani, M. (1989). Indexing strategies for goal specific retrieval of cases. Proceedings of the Case-Based Reasoning Workshop (pp. 31-35). Pensacola Beach, FL: Morgan Kaufmann.
6. Pazzani, M. (1989). Explanation-based learning with weak domain theories. Proceedings of the Sixth International Workshop on Machine Learning (pp. 72-74). Ithaca, NY: Morgan Kaufmann.
7. Sarrett, W. & Pazzani, M. (1989). One-sided algorithms for integrating empirical and explanation-based learning. Proceedings of the Sixth International Workshop on Machine Learning (pp. 26-28). Ithaca, NY: Morgan Kaufmann.
8. Schulenburg, D. & Pazzani, M. (1989). Explanation-based learning of indirect speech act interpretation rules. Proceedings of the First International Lexical Acquisition Workshop. Detroit, MI.

9. Pazzani, M. J. & Brunk, C. A. (1990). Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. Knowledge Acquisition for Knowledge-Based Systems Workshop.
10. Pazzani, M., Brunk, C. & Silverstein, G. (1991). A knowledge-intensive approach to learning relational concepts. Proceedings of the Eighth International Workshop on Machine Learning (pp. 432-436). Evanston, IL: Morgan Kaufmann.
11. Silverstein, G. & Pazzani, M. (1991). Relational clichés: Constraining constructive induction during relational learning. Proceedings of the Eighth International Workshop on Machine Learning (pp. 203-207). Evanston, IL: Morgan Kaufmann.
12. Cain, T., Pazzani, M. & Silverstein, G. (1991). Using domain knowledge to influence similarity judgments. Proceedings of the Case-Based Reasoning Workshop. Washington, DC: Morgan Kaufmann.
13. Brunk, C. & Pazzani, M. (1991). An investigation of noise tolerant relational learning algorithms. Proceedings of the Eighth International Workshop on Machine Learning (pp. 389-391). Evanston, IL: Morgan Kaufmann.
14. Murphy, P. & Pazzani, M. (1991). ID2-of-3: Constructive induction of m-of-n discriminators for decision trees. Proceedings of the Eighth International Workshop on Machine Learning (pp. 183-187). Evanston, IL: Morgan Kaufmann.
15. Hirschberg, D., Pazzani, M. & Ali, K. (1991). Average case analysis of k-CNF and k-DNF learning algorithms. Second International Workshop on Computational Learning Theory and Natural Learning Systems: Constraints and Prospects. Berkeley, CA.
16. Pazzani, M. (1992). When Prior Knowledge Hinders Learning. AAAI Workshop on Constraining learning with Prior Knowledge. San Jose, CA.
17. Murphy, P & Pazzani, M. (1993). Exploring the decision forest. Computational Learning and Natural Learning, Provincetown, MA.
18. Pazzani, M., Murphy, P., Ali, K. & Schulenburg, D. (1994). Trading off coverage for accuracy in forecasts: Applications to clinical data analysis. AAAI Symposium on AI in Medicine (pp. 106-110). Stanford, CA.
19. Ali K., Brunk C. & Pazzani M. (1995). Learning Multiple Relational Rule-based Models. In Preliminary Papers of the 5th International Workshop on Artificial Intelligence and Statistics. Fort Lauderdale, FL.
20. Pazzani, M. (1995). Searching for dependencies in Bayesian classifiers. In Preliminary Papers of the 5th International Workshop on Artificial Intelligence and Statistics. Fort Lauderdale, FL.
21. Pazzani, M., Muramatsu J. & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. AAAI Spring Symposium. Stanford, CA.
22. Starr, B. Ackerman, M. & Pazzani, M. (1996). Do I Care? -- Tell Me What's Changed on the Web. AAAI Spring Symposium. Stanford, CA.

23. R. Shankle, P. Datta & Pazzani, M. (1996). Applying machine learning to an Alzheimer's database, AAAI-96 Spring Symposium AI in Medicine: Applications of Current Technologies. Stanford, CA.
24. Merz, C. & Pazzani, M. (1997). Handling redundancy in ensembles of learned models using principal components. AAAI Workshop on Integrating Multiple Models.
25. Merz, C. & Pazzani M. (1997). Combining neural network regression estimates using principal components. Preliminary Papers of the 6th International Workshop on Artificial Intelligence and Statistics.
26. Mani, M., McDermott, S. & Pazzani, M. (1997). Generating models of mental retardation from data with machine learning Proceedings IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97), p. 114-119, IEEE Computer Society.
27. Keogh, E. & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. AAAI Workshop on Predicting the Future: AI Approaches to Time-Series Analysis. Madison, Wisc.
28. Pazzani, M. & Billsus, D. (1999). Evaluating Adaptive Web site Agents. Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval.
29. Bay, S. D. & Pazzani, M. J. (2000). Characterizing Model Performance in the Feature Space. In ICML 2000 Workshop on What Works Well Where.
30. Keogh, E., S. Chu & Pazzani, M. (2001). Using ensembles of representations for indexing large databases. International Workshop on Mining Spatial and Temporal data. In conjunction with the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-01).
31. Pazzani, M. (2005). Beyond Idiot Savants: Recommendations and Common Sense Workshop on Beyond personalization: the next stage of recommender systems research. Intelligent User Interfaces Conference, San Diego, CA
32. Michael Pazzani, Amir Feghahati, Christian Shelton, and Aaron Seitz. (2018) Explaining contrasting categories. In Workshop on Explainable Smart Systems (EXSS), ACM Intelligent User Interface Conference, Toyko, Japan.

Conference papers

1. Pazzani, M. (1983). Interactive script instantiation. Proceedings of the National Conference on Artificial Intelligence (pp. 320-326). Washington DC: Morgan Kaufmann.
2. Pazzani, M. (1984). Conceptual analysis of garden-path sentences. Proceedings of the Tenth International Conference on Computational Linguistics (pp. 486-490). Stanford, CA.
3. Pazzani, M. (1985). Explanation and generalization-based memory. Proceedings of the Seventh Annual Conference of the Cognitive Society Conference (pp. 323-328). Irvine, CA: Lawrence Erlbaum.
4. Pazzani, M. & Brindle, A. (1986). Automated diagnosis of attitude control anomalies. Proceedings of the Annual AAS Guidance and Control Conference. Keystone, CO: American Astronautical Society.

5. Pazzani, M. (1986). Refining the knowledge base of a diagnostic expert system: An application of failure-driven learning. Proceedings of the Fifth National Conference on Artificial Intelligence (pp. 1029-1035). Philadelphia, PA: Morgan Kaufmann.
6. Pazzani, M., Dyer, M. & Flowers, M. (1986). The role of prior causal theories in generalization. Proceedings of the Fifth National Conference on Artificial Intelligence (pp. 545-550). Philadelphia, PA: Morgan Kaufmann.
7. Pazzani, M., Dyer, M. & Flowers, M. (1987). Using prior learning to facilitate the learning of new causal theories. Proceedings of the Tenth International Joint Conference on Artificial Intelligence (pp. 277-279). Milan, Italy: Morgan Kaufmann.
8. Pazzani, M. & Dyer, M. (1987). A comparison of concept identification in human learning and network learning with the generalized delta rule. Proceedings of the Tenth International Joint Conference on Artificial Intelligence (pp. 147-151). Milan, Italy: Morgan Kaufmann.
9. Pazzani, M. (1988). Integrated learning with incorrect and incomplete theories. Proceedings of the Fifth International Conference on Machine Learning (pp. 291-298). Ann Arbor, MI: Morgan Kaufmann.
10. Pazzani, M. (1988). Integrating empirical and explanation-based learning methods in OCCAM. Proceedings of the Third European Working Session on Learning (pp. 147-166). Glasgow, Scotland: Pitman.
11. Pazzani, M. (1989). Learning from historical precedent. Artificial Intelligence Systems in Government Conference (pp. 150-156). Washington, DC.
12. Pazzani, M. (1989). Explanation-based learning of diagnostic heuristics: A comparison of learning from success and failure. Artificial Intelligence Systems in Government Conference (pp. 164-169). Washington DC.
13. Pazzani, M. & Schulenburg, D. (1989). The influence of prior theories on the ease of concept acquisition. Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (pp. 812-819). Ann Arbor, MI: Lawrence Erlbaum.
14. Pazzani, M. (1989). Detecting and correcting errors of omission after explanation-based learning. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (pp. 713-718). Detroit, MI: Morgan Kaufmann.
15. Pazzani, M. & Sarrett, W. (1989). Average case analysis of conjunctive learning algorithms. Proceedings of the Seventh International Conference on Machine Learning (pp. 339-347). Austin, TX: Morgan Kaufmann.
16. Pazzani, M. & Silverstein, G. (1990). Feature selection and hypothesis selection: Models of induction. Proceedings of the Twelfth Annual Conference of the Cognitive Science Society (pp. 221-228). Cambridge, MA: Lawrence Erlbaum.
17. Billman, D., Fisher, D., Gluck, M., Langley, P., Pazzani, M. (1990). Computational models of category learning. Proceedings of the Twelfth Annual Conference of the Cognitive Science Society (pp. 989-996). Cambridge, MA: Lawrence Erlbaum.

18. Hirschberg, D. & Pazzani, M. (1992). Average case analysis of k-CNF learning algorithms. Proceedings of the Tenth International Conference on Machine Learning (pp. 206-211). Aberdeen, Scotland: Morgan Kaufmann
19. Pazzani, M. & Brunk, C. (1993). Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning. The National Conference on Artificial Intelligence (pp. 328-334). Washington, D.C: AAAI Press
20. Ali, K. & Pazzani, M. (1993). HYDRA: A noise-tolerant relational concept learning algorithm. The International Joint Conference on Artificial Intelligence, Chambery, France.
21. Wogulis, J. & Pazzani, M. (1993). A methodology for evaluating theory revision systems: Results with AUDREY II. The International Joint Conference on Artificial Intelligence, Chambery, France.
22. Murphy, P. & Pazzani, M. (1994). Revision of production system rule-bases. Proceedings of the 11th International Conference of Machine Learning, New Brunswick. Morgan Kaufmann, 199-200.
23. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T. & Brunk, C. (1994). Reducing Misclassification Costs. Proceedings of the 11th International Conference of Machine Learning, New Brunswick. Morgan Kaufmann, 217-225.
24. Ali K., Brunk C. & Pazzani M. (1994). On Learning Multiple Descriptions of a Concept. In Proceedings of the Sixth International Conference on Tools with Artificial Intelligence. New Orleans, LA: IEEE Press.
25. Merz, C. & Pazzani M. (1994). Parameter Tuning for the MAX Expert System, In Proceedings of the Sixth International Conference on Tools with Artificial Intelligence. New Orleans, LA: IEEE Press. pp. 632-639.
26. Yamazaki, T., Pazzani, M & Merz, C. (1995). Learning Hierarchies from Ambiguous Natural Language Data, Proceedings of the 12th International Conference of Machine Learning.
27. Brunk, C. & Pazzani, M. (1995). A Linguistically-Based Semantic Bias for Theory Revision Proceedings of the 12th International Conference of Machine Learning.
28. Hume, T. & Pazzani, M. (1995). Learning Sets of Related Concepts: A Shared Task Model. Proceedings of the Sixteen Annual Conference of the Cognitive Science Society. Pittsburgh, PA: Lawrence Erlbaum.
29. Pazzani, M. (1995). An iterative-improvement approach for the discretization of numeric attributes in Bayesian classifiers. Proceedings of the First International Conference on Knowledge Discovery and Data Mining. Montreal: AAAI Press
30. Pazzani, M., Nguyen, L. & Mantik, S. (1995). Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent. In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence
31. Domingos, P. & Pazzani, M. (1996). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Proceedings of the International Conference on Machine Learning.
32. Pazzani, M., Muramatsu J. & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. Proceedings of the National Conference on Artificial Intelligence, Portland, OR.

33. Billsus, D. & Pazzani, M (1996). Revising user profiles: The search for interesting Web sites. International Multi-Strategy Learning Conference. Harpers Ferry, Va.
34. Pazzani, M. (1996). Constructive Induction of Cartesian Product Attributes. Information, Statistics and Induction in Science. Melbourne, Australia.
35. Starr, B., Ackerman, M. & Pazzani, M. (1996). Do-I-Care: A Collaborative Web Agent. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96), April, 1996, pp 273-274.
36. Merz, C. & Pazzani, M. (1996). Combining neural network regression estimates with regularized linear weights. Advances in Neural Information Processing Systems 9, Proceedings of the 1996 Conference. MIT Press, 564-570.
37. Pazzani, M., Mani, S. & Shankle, W. R. (1997). Beyond concise and colorful: learning intelligible rules. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA. AAAI Press, 235-238.
38. Pazzani, M., Mani, S. & Shankle, W. R. (1997). Comprehensible knowledge-discovery in databases. In M. G. Shafto & P. Langley (Ed.), Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society, pp. 596-601. Mahwah, NJ:Lawrence Erlbaum.
39. Pazzani, M., Iyer, R., See, D., Shroeder, E. & Tilles, J. (1997). CTSHIV: A Knowledge-based System in the Management of HIV-infected patients. Proceedings of the International Conference on Intelligent Information Systems.
40. Shankle, W.R., Mani, S., Pazzani, M. J. & Smyth, P. (1997). Use of a Computerized Patient Record Database of Normal Aging and Very Mildly Demented Subjects to Compare Classification Accuracies Obtained with Machine Learning Methods and Logistic Regression. Computing Science and Statistics, 29: 201-209.
41. Billsus, D. & Pazzani, M. (1998). Learning Collaborative Information Filters. Proceedings of the International Conference on Machine Learning. Morgan Kaufmann Publishers. Madison, Wisc.
42. Webb, G. & Pazzani, M. (1998). Adjusted Probability Naive Bayesian Induction. 11th Australian Joint Conference on Artificial Intelligence. Brisbane, QLD. Australia.
43. Pazzani, M. (1998). Learning with Globally Predictive Tests. The First International Conference on Discovery Science Fukuoka, Japan.
44. Mani, S. & Pazzani, M. (1998). Guideline Generation from Data by Induction of Decision Tables Using a Bayesian Network Framework JAMIA supplement p. 518-522.
45. Shankle, R., Mani, S., Dick, M & Pazzani, M. (1998). Simple Models for Estimating Dementia Severity Using Machine Learning MedInfo'98: 9th World Congress on Medical Informatics, Seoul, Korea, 1998.
46. Lathrop, R., Steffen, N., Raphael, M., Deeds-Rubin, S., Pazzani, M., Cimoch, P., See, D., Tilles, J. (1998). Knowledge-based Avoidance of Drug-Resistant HIV Mutants Proceedings of the 10th Conference on Innovative Applications of Artificial Intelligence, Madison, Wisc.

47. Keogh, E. & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Proceedings of the Fourth International Conference of Knowledge Discovery and Data Mining. pp 239-241, AAAI Press.
48. Cimoch, P.J., See, D.M., Pazzani, M.J., Reiter, W.M., Lathrop, R.H., Fasone, W.A, Tilles, J.G.; (1998). Application of a genotypic driven rule-based expert artificial intelligence computer system in treatment experienced HIV-infected patients. Immunologic and virologic response. Proc. of the 12th World AIDS Conf., Geneva, Switzerland, extended abstract #32297
49. Keogh, E. & Pazzani, M. (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. Uncertainty 99 AI and Statistics, Ft. Lauderdale, Florida, 225-230.
50. Billsus, D. & Pazzani, M. (1999). A Hybrid User Model for News Story Classification, Proceedings of the Seventh International Conference on User Modeling (UM '99), Banff, Canada.
51. Billsus, D. & Pazzani, M. (1999). A Personal News Agent that Talks, Learns and Explains, Proceedings of the Third International Conference on Autonomous Agents (Agents '99), Seattle, Washington.
52. Bay, S. D. & Pazzani, M. J. (1999). Detecting Change in Categorical Data: Mining Contrast Sets. In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining.
53. Pazzani, M. J. & Bay, S. D. (1999). The Independent Sign Bias: Gaining Insight from Multiple Linear Regression. In Proceedings of the Twenty-First Annual Meeting of the Cognitive Science Society.
54. Pazzani, M. & Billsus, D. (1999). Adaptive Web Site Agents. Proceedings of the Third International Conference on Autonomous Agents (Agents '99), Seattle, Washington.
55. Keogh, E. & Pazzani, M. (1999). Relevance Feedback Retrieval of Time Series Data The Twenty-Second Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.
56. Eamonn J. Keogh, Michael J. Pazzani: (1999). Scaling up Dynamic Time Warping to Massive Datasets. Principles and Practice of Knowledge Discovery in Databases, Prague, Czech Republic.
57. Eamonn J. Keogh, Michael J. Pazzani: (1999). An indexing scheme for similarity search in large time series databases. The 11th International Conference on Scientific and Statistical Database Management. Cleveland, Ohio.
58. S. Mani, M.B. Dick, M.J. Pazzani, E.L. Teng, D. Kempler, I.M. Taussig (1999). Refinement of Neuro-Psychological Tests for Dementia Screening in a Cross Cultural Population Using Machine Learning. Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making Aalborg, Denmark.
59. Bay, S. D. & Pazzani, M. J. (2000). Discovering and Describing Category Differences: What makes a discovered difference insightful?. Proceedings of the 22nd Annual Meeting of the Cognitive Science Society.
60. Bay, S. D. & Pazzani, M. J. (2000). Characterizing Model Errors and Differences. In Proceedings of the Seventeenth International Conference on Machine Learning.
61. Keogh, E. & Pazzani, M. (2000) Scaling up Dynamic Time Warping for Data Mining Applications. In 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, 2000.

62. Keogh, E. & Pazzani, M. (2000) A simple dimensionality reduction technique for fast similarity search in large time series databases. In the Fourth Pacific- Asia Conference on Knowledge Discovery and Data Mining. Kyoto, Japan..
63. Michael J. Pazzani: (2000) Representation of electronic mail filtering profiles: a user study. Intelligent User Interfaces 2000: 202-206
64. Daniel Billsus, Michael J. Pazzani, James Chen: (2000) A learning agent for wireless news access. Intelligent User Interfaces 2000: 33-36
65. Koji Miyahara, Michael J. Pazzani: (2000) Collaborative Filtering with the Simple Bayesian Classifier. PRICAI 2000: 679-689
66. Keogh, E., Chu, S., Hart, D. & Pazzani, M. (2001). An Online Algorithm for Segmenting Time Series. IEEE International Conference on Data Mining.
67. Keogh, E., Chu, S. & Pazzani, M. (2001). Ensemble-Index: A New Approach to Indexing Large Databases. In 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco.
68. Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra (2001). Locally adaptive dimensionality reduction for indexing large time series databases. SIGMOD 2001. Best paper award.
69. Keogh, E. & Pazzani, M. (2001). Derivative Dynamic Time Warping. In First SIAM International Conference on Data Mining (SDM'2001), Chicago, USA.
70. George Buchanan, Sarah Farrant, Matt Jones, Harold W. Thimbleby, Gary Marsden, Michael J. Pazzani: (2001) Improving mobile internet usability. WWW 10: 673-680.
71. Chu, S., Keogh, E., Hart, D. & Pazzani, M. (2002). Iterative Deepening Dynamic Time Warping for Time Series. The Second Siam International Conference on Data Mining.
72. Pazzani, M. (2002). Commercial Applications of Machine Learning for personalized wireless portals. Pacific Rim Conference on Artificial Intelligence, Springer. Pp 1-5.
73. Billsus, D., Klefstad, R & Pazzani, M.(2005). Adaptive Personalization of Mobile Content Delivery. Human Computer Interaction, Las Vegas.
74. Pazzani, M. & Brunk, C. (2005). An Improved Representation for Content-Based Television Personalization. Human Computer Interaction, Las Vegas.
75. Hettich, S & Pazzani M (2006). Mining for Proposal Reviewers: Lessons Learned at the National Science Foundation. The Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining. Philadelphia.
76. Yong Ge, Hui Xiong, Alexander Tuzhilin, Keli Xiao, Marco Gruteser, Michael J. Pazzani: (2010) An energy-efficient mobile recommender system. KDD 2010: 899-908

77. Mesterharm, C, Pazzani, M. (2011) Active learning using on-line algorithms. KDD 2011: 850-858.
78. Rawassizadeh, Reza & Dobbins, Chelsea & Nourizadeh, Manouchehr & Ghamchili, Zahra & Pazzani, Michael. (2017). A Natural Language Query Interface for Searching Personal Information on Smartwatches.. 10.1109/PERCOMW.2017.7917645.
79. Feghahati, A., Shelton, C. Pazzani, M., And Tang, K. (2020). CDeepEx: Contrastive Deep Explanations. European Conference on Artificial Intelligence.
80. Soltani, S., Kaufman, R. A., & Pazzani, M. J. (2022). User-Centric Enhancements to Explainable AI Algorithms *for Image Classification*. In *Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 44, No. 44)*.
81. 1. Pazzani, M., Soltani, S, Kauffman, R., Qian, S. Hsiao, A. (2022) Expert-Informed, User-Centric Explanations for Machine Learning, Proceedings of the AAAI Conference on Artificial Intelligence.
82. Ahamed, A . Alipour, A Kumar, K.. Soltani, S., and. Pazzani, M (2022). Improving Explanations of Image Classification with Ensembles of Learner. 8th International Conference of Artificial Intelligence and Applications. Vienna, Austria