(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 July 2025 (10.07.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/147607 A1

(51) International Patent Classification:
G06F 16/34 (2025.01) *G06F 3/01* (2006.01)
G06F 16/683 (2019.01) *G06F 40/58* (2020.01)
G10L 15/26 (2006.01)

(21) International Application Number:
PCT/US2025/010231

(22) International Filing Date:
03 January 2025 (03.01.2025)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/618,200 05 January 2024 (05.01.2024) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(72) Inventors: **BAHIRWANI, Vikas**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **XU, Susan Qi**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **FERRER GARCIA DE ALBA, Leonardo Javier**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **SMUS, Boris**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **BHATIA, Nishtha**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **LI, Lin**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **GUDA, Bhanu Prakash Reddy**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **YIN, Qiya**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **OLWAL, Alex**; GOOGLE LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(74) Agent: **SMITH, Edward P.** et al.; BRAKE HUGHES BELLERMANN LLP, P.O. Box 1077, Middletown, Maryland 21769 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA,

(54) Title: CONTEXTUAL COMPRESSION OF SPEECH

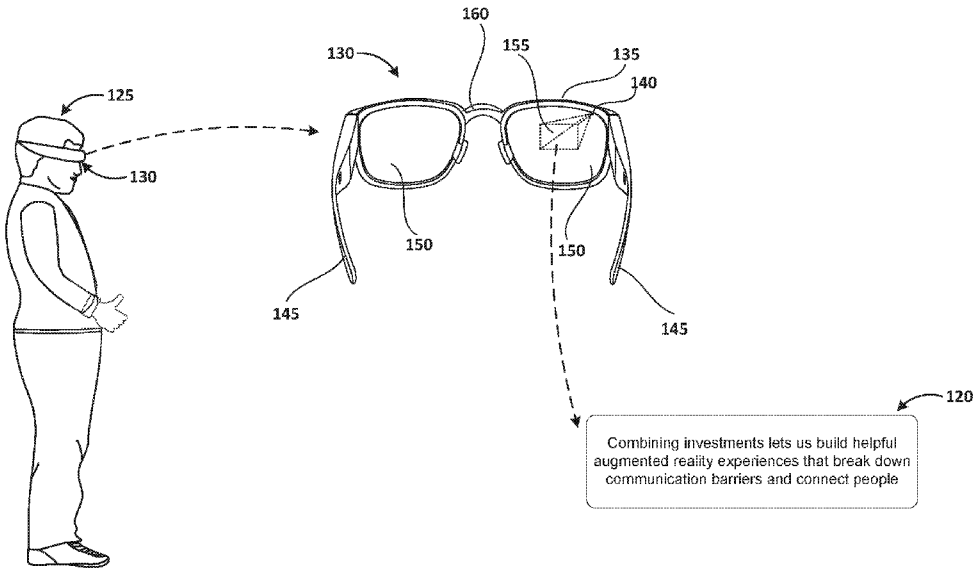


FIG. 1B

(57) Abstract: A method including generating a text packet based on speech, storing the text packet in a first queue, moving the text packet to a second queue based on a criteria, and generating compressed text representing the speech based on the text packet in the second queue.

[Continued on next page]

WO 2025/147607 A1

NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

CONTEXTUAL COMPRESSION OF SPEECH

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit and priority to U.S. Provisional Application No. 63/618,200, filed on January 5, 2024, the disclosure of which is incorporated by reference herein in its entirety.

BACKGROUND

[0002] Live captioning of spoken language can be cognitively overwhelming, and users are often seeking digestible content that enables them to stay engaged with the scene in front of them. Realtime compression (e.g., sentence compression or sentence summarization) can include compressing transcripts live during conversation to provide fluid, useful and stable transcripts to users.

SUMMARY

[0003] Some implementations relate to real-time speech compression. For example, an implementation can convert speech audio to text and store the text in a queue. When the queue includes a threshold quantity of text, the text can be compressed (or summarized) and displayed (e.g., as n lines of text) on a device (e.g., a head-worn device).

[0004] In a general aspect, a device, a system, a non-transitory computer-readable medium (having stored thereon computer executable program code which can be executed on a computer system), and/or a method can perform a process with a method including generating a text packet based on speech, storing the text packet in a first queue, moving the text packet to a second queue based on a criteria, and generating compressed text representing the speech based on the text packet in the second queue.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Example implementations will become more fully understood from the detailed description given herein below and the accompanying drawings, wherein like elements are represented by like reference numerals, which are given by way of illustration only and thus are not limiting of the example implementations.

[0006] FIG. 1A illustrates a speech or text compression flow according to at least one example implementation.

[0007] FIG. 1B illustrates a pictorial representation of an example use case of at least one example implementation.

[0008] FIG. 2 is a block diagram illustrating a dataflow for generating compressed text representing speech according to at least one example implementation.

[0009] FIG. 3 is another block diagram illustrating a dataflow for generating compressed text representing speech according to at least one example implementation.

[0010] FIG. 4 is yet another block diagram illustrating a dataflow for generating compressed text representing speech according to at least one example implementation.

[0011] FIG. 5 is still another block diagram illustrating a dataflow for generating compressed text representing speech according to at least one example implementation.

[0012] FIG. 6 illustrates a block diagram of a memory structure for generating compressed text from speech according to an example implementation.

[0013] FIG. 7 illustrates a dataflow for generating compressed text from speech according to an example implementation.

[0014] FIG. 8 illustrates a block diagram of a memory structure for generating compressed text from speech according to an example implementation.

[0015] FIG. 9 is a block diagram of a method of generating compressed text from speech according to an example implementation.

[0016] FIG. 10 illustrates a block diagram of a system for generating compressed text from speech according to an example implementation.

[0017] It should be noted that these Figures are intended to illustrate the general characteristics of methods, and/or structures utilized in certain example

implementations and to supplement the written description provided below. These drawings are not, however, to scale and may not precisely reflect the precise structural or performance characteristics of any given implementation and should not be interpreted as defining or limiting the range of values or properties encompassed by example implementations. For example, the positioning of modules and/or structural elements may be reduced or exaggerated for clarity. The use of similar or identical reference numbers in the various drawings is intended to indicate the presence of a similar or identical element or feature.

DETAILED DESCRIPTION

[0018] Text is often compressed or summarized for a variety of applications and for a variety of user situations. For example, a paragraph of a technical paper can be compressed to focus the reader on the most important information of the paragraph. Similarly, speech can be compressed to focus the listener on the most important information of the speech. Speech compression is usually compressed text that is displayed or saved for consumption by one or more users. In some applications, speech compression can often be done in real-time.

[0019] Achieving speech or text compression in real-time can include using an incremental algorithm. An incremental algorithm, such as an incremental compressor, can be a compressor that will start to generate compressed captions without waiting for the speaker to finish speaking. Some natural language processing (NLP) compression or summarization systems, however, are non-incremental. In other words, in some NLP compression or summarization systems, a full segment such as a sentence or paragraph is processed before an abstraction or summary can be generated.

[0020] At least one technical problem with the current NLP technology can be that waiting for a full segment of input before a compressed output is generated can result in unnaturally long pauses and an undesirable user experience. For example, the user experience may not be a continuous or live user experience.

[0021] For example, at least one technical problem can include passing increasingly long inputs, displaying summaries and rewriting them as more context becomes available. This NLP process can produce a non-stable output, as previously generated output may need to be partially retracted. These technical problems, such as increased latency, and instability make it difficult to apply current non-incremental models directly to achieve real-time compression since the models fail to provide true real-time experiences for users.

[0022] At least one technical solution can include an algorithm for using non-incremental models to generate a low-latency stable real-time compression. The technical solution includes dividing the technical problem into two parts. The two parts include non-incremental quality and incremental stability. The two parts help to define when to compress speech (as text) in order to strike a balance between waiting for useful information to compress and perceived end-user latency. Accordingly, the technical solution can include a first queue or stability buffer and a second queue or stable word queue. The first queue or stability buffer can be configured to store text packets representing speech until a threshold quantity of text packets or words are stored. The second queue or stable word queue can be configured to store text packets or words for compression.

[0023] At least one benefit of the technical solution can be a process for NLP that generates a low-latency stable real-time compression. The NLP process can provide a desirable user experience by generating continuous or substantially live compressed text representing real-time speech.

[0024] Several approaches for real-time NLP exist. For example, one approach is simultaneous machine translation which can provide language understanding (e.g. translation) for streaming input. However, these approaches work at the level of short text, such as partial or full sentences, and are hard to generalize to longer text meant for higher level abstraction tasks such as summarization. The regeneration of an output each time a segment is detected has been proposed for various language understanding

tasks, along with strategies to mitigate instability, but their applicability in real-time applications due to their inherent instability is limited.

[0025] On-demand compression, for example, on augmented reality (AR), mixed reality (MR), and/or extended reality (XR) devices (e.g., AR glasses) can use an "on-pause" or manual trigger-based summarization. Such approaches may not provide a fluid experience and either add cognitive load on the user to trigger summarization (on-demand) or require the speaker to artificially pause during their speech. The example implementations described herein differ in that the techniques may not use, in some implementations, a change of speaking pattern from the speaker or triggering action from the user.

[0026] Some implementations described herein can generate concise captions (e.g., compressed text) in a fluid real-time user experience that can be consumed quickly by the users on compact devices such as wearable devices, mobile devices, headsets, AR glasses, and/or the like. In at least some implementations, techniques described herein can allow the user to spend more time in the real-world connecting with the speaker live during conversations (in same or different languages).

[0027] Some implementations are configured to rewrite glitchy content into concise, simple, well-structured sentences, which can significantly improve the readability and understanding of message and information in a multilingual conversation. It is especially important for wearable devices like AR glasses, which can have limited form factors (e.g., display size) to present information. In some implementations, information dense text will be very useful to the users.

[0028] FIG. 1A illustrates a speech or text compression flow according to at least one example implementation. As shown in FIG. 1A, a speech input 105 can be used to generate a compression 110 output. In some implementations, the speech input 105 and the compression 110 output can be real-time or continuous. Over time the compression 110 output can be output as compressed text 120. In FIG. 1A, the bolded text represents newly received and/or processed speech and the greyed text represents previously received and/or processed text.

[0029] In line 115-1 speech input 105 shows a first portion of speech represented as text. The first portion of speech is shown as not being compressed in compression 110. The first portion of speech may not include sufficient text for compression according to an example implementation (described in more detail below). In line 115-2 speech input 105 shows a second portion of speech concatenated to the first portion represented as text. In line 115-2 compression 110 includes compressed text representing the first portion of speech and the second portion of speech.

[0030] In line 115-3 speech input 105 shows a third portion of speech concatenated to the first portion and the second portion represented as text. In line 115-3 compression 110 includes compressed text representing the first portion of speech, the second portion of speech and the third portion of speech. In line 115-4 speech input 105 shows a fourth portion of speech concatenated to the first portion, the second, and the third portion represented as text. In line 115-4 compression 110 includes compressed text representing the first portion of speech, the second portion of speech, the third portion of speech, and the fourth portion of speech.

[0031] In some implementations, any NLP model (e.g., sequence to sequence models, large language model (LLM), and the like) can be used to generate the compressed text shown in compression 110. Some implementations can include the use of models that can, at least, produce high quality compression of static input text, can benefit from additional input (e.g., previous compression, previous input, and the like), and/or produce monotonically increasing outputs. In other words, line 115-3 can use (or benefit from) at least one of line 115-1 and/or line 115-2.

[0032] In some implementations, monotonicity can refer to the characteristic of the model to produce an output that is an append output or an append only output. In other words, in some implementations, given an input that is additional to the previous input (of the previous call to the algorithm), the output of the model does not change previously produced output (or a previous compression 110) and adds or only adds to previously produced output (or a previous compression 110). In some implementations, appending outputs can ensure that compressions do not visually flicker and are stable.

FIG. 1A illustrates at least one example of monotonous compression. In some implementations, models with lower latencies can be used to keep the end-user latencies to a minimum.

[0033] FIG. 1B illustrates a pictorial representation of an example use case of at least one example implementation. FIG. 1B illustrates a head-worn device (e.g., AR glasses) use case of at least one example implementation. As shown in FIG. 1B, a user 125 is wearing a head-worn device 130. In some implementations, the head-worn device 130 can include AR glasses. In some implementations, compressed text 120 can be displayed or rendered on a display of the head-worn device 130. In some implementations, the display of the head-worn device 130 may be capable of displaying a limited amount of text and/or a limited number of lines of text. For example, the display of the head-worn device 130 may be capable of displaying a maximum of five (5) lines of text. Therefore, in some implementations the most recent (e.g., five (5) most recent lines) of compressed text 120 may be displayed or rendered on the display of the head-worn device 130.

[0034] As shown in FIG. 1B, the example head-worn device 130 includes a frame 135 with a bridge 160 connecting two portions of the frame 135. The frame 135 includes a front frame portion defined by two portions surrounding respective optical portions in the form of lenses 150, with a bridge portion 160 connecting the two portions. Arm portions 145 are coupled, for example, pivotably or rotatably coupled, to the front frame by a hinge. In some examples, the lenses 150 may be corrective/prescription lenses. In some examples, the lenses 150 may be an optical material including glass and/or plastic portions that do not necessarily incorporate corrective/prescription parameters.

[0035] A display device 140 may be coupled in a portion of the frame 135. In the example shown in FIG. 1B, the display device 140 is coupled in the arm portion 145 of the frame 135. In some examples, the head-worn device 130 can also include an audio output device (such as, for example, one or more speakers), an illumination device, a sensing system, a control system, at least one processor, and a camera (all not shown for clarity). In some examples, the display device 140 may include a see-through near-

eye display. For example, the display device 140 may be configured to project light from a display source onto a portion of teleprompter glass functioning as a beamsplitter seated at an angle (e.g., 30-45 degrees).

[0036] The beamsplitter may allow for reflection and transmission values that allow the light from the display source to be partially reflected while the remaining light is transmitted through. Such an optic design may allow a user to see both physical items in the world, for example, through the lenses 150, next to content (for example, digital images, user interface elements, virtual content, and the like) generated by the display device 140. In some implementations, waveguide optics may be used to depict content on display device 140 via outcoupled light 155 which can be used to display or render compressed text 120.

[0037] FIGS. 2-5 are block diagrams illustrating dataflows for generating compressed text representing speech according to at least one example implementation. As shown in FIG. 2, the dataflow includes a microphone 205, a speech to text module 210, a queue 215, a queue 220, a text compression module 225, and a display 230.

[0038] The microphone 205 can be configured to capture audio and convert the audio to electric signals. In some implementations, the audio signals represent speech. The microphone 205 can be an element of a device (e.g., head-worn device 130). The microphone 205 can be a standalone device. The microphone 205 can be any type of microphone (e.g., a piezoelectric device).

[0039] The speech to text module 210 can be configured to generate text representing speech based on electric signals received from microphone 205. In some implementations, the speech to text module 210 can be configured to generate individual words representing speech (or a portion of speech). In some implementations, the speech to text module 210 can be configured to generate a text packet representing speech (or a portion of speech). In some implementations, the text packet can include a plurality of words representing speech (or a portion of speech).

[0040] In some implementations, the speech to text module 210 can use a machine learned model trained to recognize speech (e.g., audio and language) and transcribe speech. In some implementations audio representing speech can be received. The audio (or audio data) can be communicated to an application programming interface (API) to recognize and transcribe the speech. In some implementations, the speech-to-text process can be performed as synchronous, asynchronous, and/or streaming speech-to-text recognition. Synchronous recognition requests can be short time frames (e.g., limited to audio data of 1 minute or less in duration). Asynchronous recognition requests can be longer time frames (e.g., limited to audio data of 500 minutes or less in duration).

[0041] Streaming recognition can be configured to provide interim results while audio is being captured, allowing a result to appear, for example, while a user is still speaking. Accordingly, some implementations of the speech to text module 210 can use a streaming recognition procedure. In some implementations, streaming audio recognition generates individual words and then generates a text packet including a plurality of words.

[0042] In some implementations, a streaming Speech-to-Text API recognition model can be configured for real-time capture and recognition of audio, within a bi-directional stream. Accordingly, the speech to text module 210 can send audio (e.g., to a web-based text recognition server) on a request stream and receive interim and final recognition results on a response stream in real time. Interim results represent the current recognition result for a section of audio, while the final recognition result represents the last, best guess for that section of audio. In some implementations, the speech to text module 210 can be configured to send the interim results to queue 215. In some implementations, the speech to text module 210 can be configured to send the final recognition result(s) to queue 215. In implementations, the speech to text module 210 can be configured to send the interim results and the final recognition result(s) to queue 215.

[0043] The queue 215 can be configured to store text packets. In some implementations, the text packet represents speech (or a portion of speech). In some implementations, the text packet can include a plurality of words representing speech (or a portion of speech). In some implementations, queue 215 can be configured to receive a text packet, extract words from the text packet and store the words. In some implementations, queue 215 can be referred to as a stability buffer.

[0044] As the queue 215 receives text packets that represent speech, queue 215 can store text until a criteria (or criterion) is satisfied. Then, text packet(s), or the words included in the text packets, can be moved to the queue 220 as stable words. Moving text packet(s) can use a first in – first out (FIFO) algorithm. The criteria can be based on a number of words. For example, when M words are in queue 215, a text packet(s), word, and/or words can be moved from the queue 215 to queue 220. The text packet(s), word, and/or words in queue 220 can then be compressed.

[0045] In some implementations, each incoming text packet can be converted into words. At a given time, all (or at least some) of the words older than M words will be considered stable and input for compression. In some implementations, the queue 215 or stability buffer will handle de-duplication of the words automatically by always considering the latest packets as the queue 215 will ensure that packets older than M do not change.

[0046] In some implementations, the queue 215 or stability buffer size M (in number of words) can be set to any number. In some implementations, the larger the number, the more the user will have to wait for the compression to be triggered. In some implementations, the smaller the number the more unstable the input transcript. In some implementations, this can be evaluated through representative examples and set as a parameter before the algorithm begins. Note that if there is no instability in the ASR, this step may be skipped by setting stability buffer size to zero (0).

[0047] In some implementations the size of the queue 215 or stability buffer, M, can be a variable number. In some implementations, the size of the queue 215 can be set at the beginning of a compression process. In some implementations, the size of the

queue 215 can change during a compression process. For example, the stability of the compression process can be monitored. Should stability satisfy a criteria, the size of the queue 215 can remain constant. Should stability not satisfy a criteria, the size of the queue 215 can be changed. For example, if stability is below a threshold, the size of the queue 215 can be increased to increase stability. For example, if stability is above a threshold, the size of the queue 215 can be decreased to decrease latency.

[0048] The queue 220 can be configured to store words representing speech. In some implementations, queue 220 can be configured to store stable words. In some implementations, queue 220 can be configured to store words representing speech that are in a condition to be compressed. In some implementations, the model can be any NLP model (e.g., Sequence to Sequence models, large language model (LLM), and the like) that can be used to generate compressed text (e.g., compression 110). In some implementations, the model can, at least, produce high quality compression of static input text, can benefit from additional input (e.g., previous compression, previous input, and the like), and produce monotonically increasing outputs.

[0049] In some implementations, queue 215 can be referred to as a stability buffer. In some implementations, queue 220 can be referred to as a stable words buffer. The stability buffer (queue, first queue, and the like) can be configured to store words representing speech. The stability buffer can be configured to store a first quantity (e.g., a threshold quantity) of words. Then, when the first quantity of words is reached, a second quantity (e.g., a threshold quantity) of words is communicated to the stable words buffer and the communicated words are deleted from the stability buffer. Accordingly, the queue 215 or stability buffer can be configured to temporarily store words representing speech.

[0050] The stable words buffer (queue, second queue, and the like) can be configured to store words representing speech. The stable words buffer can be configured to store words as the words representing speech are compressed. In some implementations, as the words representing speech are compressed, the words remain in the stable words buffer. In other words, words are not removed from stable words

buffer as the words representing speech are compressed. In some implementations, the stable words buffer can receive a plurality of words from the stability buffer. Receiving the words can trigger the stable words buffer to send the received words to the text compression module 225. In addition to sending the received words, the stable words buffer can also send previously compressed words to the text compression module 225.

[0051] Accordingly, queue 215 (e.g., stability buffer) and queue 220 (e.g., stable words buffer) perform different functions. Queue 215 and queue 220 do not just store words. Queue 215 stores words until the words are determined to be ready (e.g., a threshold quantity of words are in the queue) to be compressed, communicates the words (e.g., to queue 220), and removes the communicated words from queue 220. Queue 220 stores words that are ready to be compressed and words that have previously been compressed. In addition, queue 220 triggers a compression operation when new words are received. Alternatively, a processor triggers a compression operation when new words are received by queue 220. Queue 220 can be configured to communicate the new words and the previously compressed words when the compression operation is triggered.

[0052] In some implementations, a first queue can be configured to store words. In some implementations, the first queue can be configured to determine the words are to be compressed. In some implementations, the first queue can be configured to, communicate the words to a second queue. In some implementations, the first queue can be configured to remove the communicated words from the first queue. In some implementations, a second queue can be configured to store words to be compressed. In some implementations, the second queue can be configured to store words that have previously been compressed. In some implementations, a method can include determining the first queue includes a threshold quantity of words. In some implementations, in response to determining the first queue includes the threshold quantity of words a method can include communicating a portion of the words to a second queue. In some implementations, a method can include removing the portion

of the words from the first queue. In some implementations, a method can include receiving, by the second queue from the first queue, the portion of the words. In some implementations, in response to receiving the portion of the words, a method can include causing the portion of the words to be compressed. In some implementations, causing the portion of the words to be compressed can include communicating the portion of the words. In some implementations, causing the portion of the words to be compressed can include communicating previously compressed words to a compression module.

[0053] The text compression module 225 can be configured to compress words representing speech (e.g., see compression 110 and compressed text 120). In some implementations, the compression module 225 can include a compression model. The compression model can be configured to compress words representing speech. In some implementations, the model can be a non-incremental model configured to perform a low-latency stable real-time compression of text representing speech.

[0054] Text compression can use a machine-learned model configured to distill text, documents, or content into a short and easily digestible format. For example, the model can use natural language processing or understanding to condense a long stream of words and restate its most important takeaways in just a few sentences.

[0055] The display 230 can be configured to display compressed text (e.g., compressed text 120). In some implementations, the display 230 can be a component of, for example, a computing device, a wearable device, a head-worn device, an XR device, smart glasses, a mobile computing device, a tablet computing device, a laptop, and the like).

[0056] In some implementations, the model used by the text compression module 225 may be configured to compress text representing speech in a limited number of languages. For example, the text compression module 225 may be configured to compress text in English, Spanish, and French. However, a speaker (via microphone 205) may be speaking in a language not recognized by the model used by the text compression module 225. For example, the speaker may be speaking German or

Chinese. Therefore, the text generated by the speech to text module 210 may need to be translated into a language recognized by the model used by the text compression module 225. For example, the text generated by the speech to text module 210 may need to be translated from German into English. As shown in FIG. 3, the dataflow includes the microphone 205, the speech to text module 210, a translation module 305, the queue 215, the queue 220, the text compression module 225, and the display 230.

[0057] The translation module 305 can be configured to translate a word(s) from a first language to a second language. In some implementations, the translation module 305 can be configured to translate a word(s) from a first language to a second language using a translation algorithm. In some implementations, the translated word(s) can represent speech. In some implementations, the translation module 305 can be configured to translate a word(s) into a language recognized by the model used by the text compression module 225.

[0058] In some implementations, the model used by the text compression module 225 may be configured to compress text representing speech in a limited number of languages. For example, the text compression module 225 may be configured to compress text in English, Spanish, and French. However, a reader of the compressed text (via display 230) may read a language not recognized by the model used by the text compression module 225. For example, the reader may read German or Chinese. Therefore, the compressed text generated by the text compression module 225 may need to be translated into a language recognized by the reader of the compressed text. For example, the text generated by the text compression module 225 may need to be translated from English into German. As shown in FIG. 4, the dataflow includes the microphone 205, the speech to text module 210, the queue 215, the queue 220, the text compression module 225, a translation module 405, and the display 230.

[0059] The translation module 405 can be configured to translate a word(s) from a first language to a second language. In some implementations, the translation module 405 can be configured to translate a word(s) from a first language to a second

language using a translation algorithm. In some implementations, the translated word(s) can represent speech. In some implementations, the translation module 405 can be configured to translate a word(s) into a language recognized by the user reading text using the display 230.

[0060] In some implementations, the model used by the text compression module 225 may be configured to compress text representing speech in a limited number of languages. For example, the text compression module 225 may be configured to compress text in English, Spanish, and French. However, a speaker (via microphone 205) may be speaking in a language not recognized by the model used by the text compression module 225. For example, the speaker may be speaking German or Chinese. Therefore, the text generated by the speech to text module 210 may need to be translated into a language recognized by the model used by the text compression module 225. For example, the text generated by the speech to text module 210 may need to be translated from German into English.

[0061] Further, in some implementations, the model used by the text compression module 225 may be configured to compress text representing speech in a limited number of languages. For example, the text compression module 225 may be configured to compress text in English, Spanish, and French. However, a reader of the compressed text (via display 230) may read a language not recognized by the model used by the text compression module 225. For example, the reader may read German or Chinese. Therefore, the compressed text generated by the text compression module 225 may need to be translated into a language recognized by the reader of the compressed text. For example, the text generated by the text compression module 225 may need to be translated from English into Chinese. As shown in FIG. 5, the dataflow includes the microphone 205, the speech to text module 210, the translation module 305, the queue 215, the queue 220, the text compression module 225, a translation module 405, and the display 230.

[0062] In some implementations, it can be important for the compression algorithm to operate on a stable input transcript to ensure (or at least empower the model for) the

generation of stable compression downstream based on the same. In some implementations, latency can be decreased to provide a live fluid experience to the users. In some implementations, a simple yet powerful technique of a 'Stability buffer' can be implemented. FIG. 6 illustrates a block diagram of a memory structure for generating compressed text from speech according to an example implementation.

[0063] As shown in FIG. 6, the queue 215 can be referred to as a stability buffer and the queue 220 can include stable words. As the queue 215 receives text packets that represent speech, queue 215 can store text until a criteria is satisfied. Then, text packet(s), or the words included in the text packets, can be moved to the queue 220 as stable words. Moving text packet(s) can use a first in – first out (FIFO) algorithm. The criteria can be based on a number of words. Alternatively, or in addition, the criteria can be based on an amount of time that words are in the buffer. For example, when M words are in queue 215 (and/or M words are in queue 215 for t seconds), a text packet(s), word, and/or words can be moved from the queue 215 to the queue 220. The text packet(s), word, and/or words in the queue 220 can then be compressed.

[0064] In some implementations, each incoming text packet can be converted into words. At a given time, all (or at least some) of the words older than M words will be considered stable and input for compression. In some implementations, the queue 215 or stability buffer will handle de-duplication of the words automatically by always considering the latest packets as the queue 215 will ensure that packets older than M do not change.

[0065] In some implementations, the queue 215 or stability buffer size M (in number of words) can be set to any number. In some implementations, the larger the number, the more the user will have to wait for the compression to be triggered. In some implementations, the smaller the number the more unstable the input transcript. In some implementations, this can be evaluated through representative examples and set as a parameter before the algorithm begins. Note that if there is no instability in the ASR, this step may be skipped by setting stability buffer size to zero (0).

[0066] In some implementations the size of the queue 215 or stability buffer, M, can be a variable number. In some implementations, the size of the queue 215 can be set at the beginning of a compression process. In some implementations, the size of the queue 215 can change during a compression process. For example, the stability of the compression process can be monitored. Should stability satisfy a criteria, the size of the queue 215 can remain constant. Should stability not satisfy a criteria, the size of the queue 215 can be changed. For example, if stability is below a threshold, the size of the queue 215 can be increased to increase stability. For example, if stability is above a threshold, the size of the queue 215 can be decreased to decrease latency.

[0067] FIG. 7 illustrates a dataflow for generating compressed text from speech according to an example implementation. The system includes a client 705 and a model 710. In some implementations, the client 705 and the model 710 can be in a same device. In some implementations, the client 705 and the model 710 can be in different devices. The client 705 can be configured to manage the buffering, storage, communication, and further processing of text representing speech. Model 710 can be configured to compress text representing speech. In some implementations, the client 705 and the model 710 (together) can be configured to implement the techniques (e.g., algorithms) described herein.

[0068] As shown in FIG. 7, the input to at least some algorithms described herein is text. Specifically, text that is output from, for example, an automatic speech recognition (ASR) system. The speaker's voice, in some implementations, can be converted into text and passed to implementations described herein for processing. Additionally, to ensure monotonicity of the output, previous context such as previous compression, previous few lines, first few lines, initial compression, conversation topic may also be provided to some implementations.

[0069] The output of some implementations can be text as well. In some implementations, this output is the compressed version of the input text provided. Some implementations can be language agnostic and as such, can handle inputs and outputs from any natural language. In some implementations, the output will always be in the

same language as the input. In some implementations, if output in a different language is desired, it may simply be translated via a translation algorithm.

[0070] FIG. 8 illustrates a block diagram of a memory structure for generating compressed text from speech according to an example implementation. In some implementations, the memory structure can be used to implement the techniques (e.g., algorithms) described herein. In FIG. 8, X can represent input text packets of size K words and $C_{x-2, x-1}$ can represent a compression based on input [x-2, x-1].

[0071] In some implementations, incremental algorithms can use non-incremental models by repeatedly feeding the model longer inputs and inferring outputs. In some implementations, the length (or the information content) of the input can become too large for the model to handle and the model produces degenerate outputs (e.g. empty compressions). In some implementations, when that happens, the model is said to have stagnated. In some implementations, the algorithm can then restart the incremental process by providing a shorter input (consisting of only the most recent input content). In some implementations, the restart can occur after a number of stagnations. The number of stagnations before restarting can be a variable number or a predefined number of stagnations. In some implementations a counter is incremented each time stagnation occurs. In some implementations, the counter is reset or set to zero (0) each time a compression is completed successfully (e.g., compressed text is generated).

[0072] In some implementations, as seen in FIG. 8, compression C_{X-2} is based on input [X-2]. Compression $C_{X-2, X-1}$ is based on progressively concatenated inputs [X-2, X-1]. In some implementations, when compressing for concatenated input [X-2, X-1, X], the model stagnates and does not produce any additional compression. Thus, while making the next call, some implementations can be configured to restart incrementality and request compression only on the stagnated input packet X and the new input packet X+1 (e.g., [X, X+1] and not [X-2, X-1, X, X+1]).

[0073] It is worth noting that an implementation of this scheme may have made a compression call for X only. However, in some implementations, this call would have caused additional delay in compressing X due to stagnation. In some implementations,

in a live conversation this can add extra latency to the user interface. It is up to specific implementation to either call compression for X or [X, X+1] if X+1 has already arrived.

[0074] In some implementations, restarting incrementality every time the model stagnates may not be ideal, because it may not have enough information in the subsequent packets to compress anything more than what it already has. In some implementations, waiting for multiple stagnations before restarting the model can ensure that stagnation has occurred due to model inability. In some implementations, waiting for 2 - 3 stagnations may be sufficient to identify the need for a restart. In some implementations, this parameter is configurable through experimentation and can be set before the algorithm executes.

[0075] In some implementations, while lengthy inputs can cause the model to stagnate, extremely short inputs or tiny updates to the input transcript can also lead the model to not produce any new compression in addition to what it already has. Thus, before making a compression call, some implementations are configured to wait for at least K stable words. As seen in at least FIG. 8, each input X (X-2, X-1, X.... X+2 etc.) is a group of K words. Some implementations are configured to issue compression requests at every K set of words. Determining when to compress can be based on, for example, compressing sentence boundaries, compressing on a word-based sliding window (Window-K as opposed to Wait-K), compressing on wall-clock based sliding window, compressing after a certain set of part-of-speech (POS) tags have been discovered (e.g. Noun, verb, or subject verb object etc.), and the like.

[0076] Below is an example of a compression algorithm.

1. As the session starts, begin buffering words in the stability buffer of size M words.
2. Once the stability buffer is full, start moving the words to a 'Stable Words' collection in the First in First Out (FIFO) paradigm.
3. Create compression input packets Xi from every K words in the "Stable Words" collection.
4. Manage variables:
    a. PreviousContext = ""

b. Input = ‘’’’
c. StagnationCount = 0

5. For every input packet Xi:
    a. Input += Xi
    b. Ci = Compress (Input, PreviousContext)
    c. If Ci is not empty:
        i. Display Ci
        ii. PreviousContext += Ci
        iii. StagnationCount = 0
    d. Else
        i. StagnationCount += 1
    e. If StagnationCount >= StagnationThreshold:
        i. PreviousContext = “”
        ii. Input = [Xi-StagnationCount-1 …Xi]
6. If there is a significant pause in the speech (managed by inactivity threshold), prepare input packets X with remaining words in the “Stable Words” collection regardless of K and go to Step 5.

[0077] At least one feature worth noting is that the compression model may need at least two inputs (a) progressively appended inputs (Input = Input + X) for incrementality, and (b) PreviousContext (PreviousContext = PreviousContext + Ci) for monotonicity. In some implementations, previous context may be one or more of previous compression, previous few lines, first few lines, initial compression, conversation topic.

[0078] In some implementations, this technique could be valuable to use in various products where improving speech-to-text is helpful and where high quality summaries would be beneficial. Beyond head-worn displays, wearables, and mobile devices, the technology could also be used in remote conferencing and web apps to, for example, improve translation and compression / summarization results.

[0079] Some implementations of the concepts may be applied to a variety of products that require streaming language understanding in real-time. Examples include summarization or simplification of spoken language on mobile devices. Additionally, many niche players in the video conferencing, remote meeting space might find this technology relevant.

[0080] Incrementality - Processing the input even before the sentence has been completed. Incremental models work on 'partial' inputs and produce 'partial outputs'. Non-incremental models take 'final' inputs and produce 'final' or 'non-incremental' outputs.

[0081] Stability - A stable output is displayed or produced monotonically. An unstable output is where the model retracts pieces of previous outputs produced leading to flicker.

[0082] Example 1. FIG. 9 is a block diagram of a method of generating compressed text from speech according to an example implementation. As shown in FIG. 9, in step S905 generate a (and/or at least one) text packet based on speech. In step S910 store the text packet in a first queue. In step S915 move the text packet to a second queue based on a criteria. In step S920 generate compressed text representing the speech based on the text packet in the second queue.

[0083] The criteria can be based on a threshold amount of data. The criteria can be based on a threshold number of text packets. The criteria can be based on a threshold number of words. The criteria can be based on a threshold time. The criteria can be based on a threshold quantity of time. The criteria can be based on a threshold time lapse.

[0084] In some implementations, the moving of a text packet to a second queue may be based on, for example, an analysis of the data contained in the first queue. In some implementations, the moving of a text packet to a second queue may be based on the amount of data contained in the first queue, e.g., based on the amount of data contained in the first queue exceeding a threshold value and/or threshold quantity. In some implementations, the moving of a text packet to a second queue may be based on the total size of the text packet or text packets contained in the first queue, e.g., based on the total size of the text packet or text packets contained in the first queue exceeding a threshold value and/or threshold quantity. In some implementations, the moving of a text packet to a second queue may be based on the total number of words or words contained in the first queue, e.g., based on the total number of words or words contained

in the first queue exceeding a threshold value and/or threshold quantity. In some implementations, the moving of a text packet to a second queue may be based on the time or amount of time the text packets contained in the first queue, e.g., based on the time or amount of time the text packets contained in the first queue exceeding a threshold value, a threshold time lapse, and/or threshold time.

[0085] Example 2. The method of Example 1, wherein the second queue can include a plurality of text packets and the generating of the compressed text representing the speech can be based on the plurality of text packets.

[0086] Example 3. The method of Example 1 can further include identifying a quantity of stagnated text and in response to determining that the quantity of stagnated text satisfies a criteria, restarting the generating of the compressed text.

[0087] Example 4. The method of Example 1, wherein the text packet can include a predefined number of words.

[0088] Example 5. The method of Example 1 can further include displaying the compressed text on a wearable device.

[0089] Example 6. The method of Example 1 can further include translating text associated with the text packet from a first language to a second language.

[0090] Example 7. The method of Example 1 can further include translating compressed text from a first language to a second language.

[0091] Example 8. The method of Example 1, wherein generating the text packet can includes receiving a stream of audio representing speech, generating text based on the stream of audio, generating a first text packet including a first portion of the generated text, and generating a second text packet including the first portion and a second portion of the generated text.

[0092] Example 9. The method of Example 1 can further include detecting a pause in the speech and in response to detecting the pause in the speech, ending the generating of the compressed text representing the speech.

[0093] Example 10. The method of Example 9, wherein the pause in the speech can be detected based on a threshold amount of time without speech.

[0094] Example 11. A method can include any combination of one or more of Example 1 to Example 10.

[0095] Example 12. A non-transitory computer-readable storage medium comprising instructions stored thereon that, when executed by at least one processor, are configured to cause a computing system to perform the method of any of Examples 1-11.

[0096] Example 13. An apparatus comprising means for performing the method of any of Examples 1-11.

[0097] Example 14. An apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus at least to perform the method of any of Examples 1-11.

[0098] FIG. 10 illustrates a block diagram of a system for generating compressed text from speech according to an example implementation. As shown in FIG. 10, the system can include at least one processor 1005, at least one memory 1010, the speech to text module 210, the queue 215, the queue 220, the text compression module 225 (including a compression model 1015), the translation module 305, and the translation module 405. The processor 1005 and the memory 1010 are communicatively coupled via bus 1020. In some implementations, the system can be included in a computing device (e.g., head-worn device 130).

[0099] The at least one processor 1005 may be utilized to execute instructions stored on the at least one memory 1010, so as to thereby implement the various features and functions described herein, or additional or alternative features and functions. The at least one processor 1005 and the at least one memory 1010 may be utilized for various other purposes. In particular, the at least one memory 1010 can represent an example of various types of memory and related hardware and software which might be used to implement any one of the modules described herein.

[00100] The at least one memory 1010 may be configured to store data and/or information associated with a text compression system or device. For example, the at least one memory 1010 may be configured to store codecs associated with encoding

images and/or algorithms for selecting a portion of an image. For example, the at least one memory 1010 may be configured to store code associated with selecting a portion of an image based on available system resources. The at least one memory 1010 may be a shared resource. For example, the text compression system or device may be an element of a larger system (e.g., a personal computer, a mobile device, a wearable device, and the like). Therefore, the at least one memory 1010 may be configured to store data and/or information associated with other elements within the larger system.

[00101] Example implementations can include a non-transitory computer-readable storage medium comprising instructions stored thereon that, when executed by at least one processor, are configured to cause a computing system to perform any of the methods described above. Example implementations can include an apparatus including means for performing any of the methods described above. Example implementations can include an apparatus including at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus at least to perform any of the methods described above.

[00102] Various implementations of the systems and techniques described here can be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations can include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

[00103] These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and can be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms “machine-

readable medium" "computer-readable medium" refers to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term "machine-readable signal" refers to any signal used to provide machine instructions and/or data to a programmable processor.

[00104] To provide for interaction with a user, the systems and techniques described here can be implemented on a computer having a display device (a LED (light-emitting diode), or OLED (organic LED), or LCD (liquid crystal display) monitor/screen) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user can be received in any form, including acoustic, speech, or tactile input.

[00105] The systems and techniques described here can be implemented in a computing system that includes a back end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front end component (e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the systems and techniques described here), or any combination of such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

[00106] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of

computer programs running on the respective computers and having a client-server relationship to each other.

[00107] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the specification.

[00108] In addition, the logic flows depicted in the figures do not require the particular order shown, or sequential order, to achieve desirable results. In addition, other steps may be provided, or steps may be eliminated, from the described flows, and other components may be added to, or removed from, the described systems. Accordingly, other implementations are within the scope of the following claims.

[00109] While certain features of the described implementations have been illustrated as described herein, many modifications, substitutions, changes and equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the scope of the implementations. It should be understood that they have been presented by way of example only, not limitation, and various changes in form and details may be made. Any portion of the apparatus and/or methods described herein may be combined in any combination, except mutually exclusive combinations. The implementations described herein can include various combinations and/or sub-combinations of the functions, components and/or features of the different implementations described.

[00110] While example implementations may include various modifications and alternative forms, implementations thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that there is no intent to limit example implementations to the particular forms disclosed, but on the contrary, example implementations are to cover all modifications, equivalents, and alternatives falling within the scope of the claims. Like numbers refer to like elements throughout the description of the figures.

[00111] Some of the above example implementations are described as processes or methods depicted as flowcharts. Although the flowcharts describe the operations as sequential processes, many of the operations may be performed in parallel, concurrently or simultaneously. In addition, the order of operations may be re-arranged. The processes may be terminated when their operations are completed, but may also have additional steps not included in the figure. The processes may correspond to methods, functions, procedures, subroutines, subprograms, etc.

[00112] Methods discussed above, some of which are illustrated by the flow charts, may be implemented by hardware, software, firmware, middleware, microcode, hardware description languages, or any combination thereof. When implemented in software, firmware, middleware or microcode, the program code or code segments to perform the necessary tasks may be stored in a machine or computer readable medium such as a storage medium. A processor(s) may perform the necessary tasks.

[00113] Specific structural and functional details disclosed herein are merely representative for purposes of describing example implementations. Example implementations, however, be embodied in many alternate forms and should not be construed as limited to only the implementations set forth herein.

[00114] It will be understood that, although the terms first, second, etc. may be used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another. For example, a first element could be termed a second element, and, similarly, a second element could be termed a first element, without departing from the scope of example implementations. As used herein, the term and/or includes any and all combinations of one or more of the associated listed items.

[00115] It will be understood that when an element is referred to as being connected or coupled to another element, it can be directly connected or coupled to the other element or intervening elements may be present. In contrast, when an element is referred to as being directly connected or directly coupled to another element, there are no intervening elements present. Other words used to describe the relationship between

elements should be interpreted in a like fashion (*e.g.*, between versus directly between, adjacent versus directly adjacent, etc.).

[00116] The terminology used herein is for the purpose of describing particular implementations only and is not intended to be limiting of example implementations. As used herein, the singular forms a, an and the are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms comprises, comprising, includes and/or including, when used herein, specify the presence of stated features, integers, steps, operations, elements and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components and/or groups thereof.

[00117] It should also be noted that in some alternative implementations, the functions/acts noted may occur out of the order noted in the figures. For example, two figures shown in succession may in fact be executed concurrently or may sometimes be executed in the reverse order, depending upon the functionality/acts involved.

[00118] Unless otherwise defined, all terms (including technical and scientific terms) used herein have the same meaning as commonly understood by one of ordinary skill in the art to which example implementations belong. It will be further understood that terms, e.g., those defined in commonly used dictionaries, should be interpreted as having a meaning that is consistent with their meaning in the context of the relevant art and will not be interpreted in an idealized or overly formal sense unless expressly so defined herein.

[00119] Portions of the above example implementations and corresponding detailed description are presented in terms of software, or algorithms and symbolic representations of operation on data bits within a computer memory. These descriptions and representations are the ones by which those of ordinary skill in the art effectively convey the substance of their work to others of ordinary skill in the art. An algorithm, as the term is used here, and as it is used generally, is conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical

manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of optical, electrical, or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[00120] In the above illustrative implementations, reference to acts and symbolic representations of operations (e.g., in the form of flowcharts) that may be implemented as program modules or functional processes include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types and may be described and/or implemented using existing hardware at existing structural elements. Such existing hardware may include one or more Central Processing Units (CPUs), digital signal processors (DSPs), application-specific-integrated-circuits, field programmable gate arrays (FPGAs) computers or the like.

[00121] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, or as is apparent from the discussion, terms such as processing or computing or calculating or determining of displaying or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical, electronic quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[00122] Note also that the software implemented aspects of the example implementations are typically encoded on some form of non-transitory program storage medium or implemented over some type of transmission medium. The program storage medium may be magnetic (e.g., a floppy disk or a hard drive) or optical (e.g., a compact disk read only memory, or CD ROM), and may be read only or random access.

Similarly, the transmission medium may be twisted wire pairs, coaxial cable, optical fiber, or some other suitable transmission medium known to the art. The example implementations are not limited by these aspects of any given implementation.

[00123] Lastly, it should also be noted that whilst the accompanying claims set out particular combinations of features described herein, the scope of the present disclosure is not limited to the particular combinations hereafter claimed, but instead extends to encompass any combination of features or implementations herein disclosed irrespective of whether or not that particular combination has been specifically enumerated in the accompanying claims at this time.

WHAT IS CLAIMED IS:

1. A method comprising:

generating a text packet based on speech;

storing the text packet in a first queue;

moving the text packet to a second queue based on a criteria; and

generating compressed text representing the speech based on the text packet in the second queue.

2. The method of claim 1, wherein

the second queue includes a plurality of text packets, and

the generating of the compressed text representing the speech is based on the plurality of text packets.

3. The method of claim 1 or 2, further comprising:

identifying a quantity of stagnated text; and

in response to determining that the quantity of stagnated text satisfies a criteria, restart the generating of the compressed text.

4. The method of any of claim 1 to claim 3, wherein the text packet includes a predefined number of words.

5. The method of any of claim 1 to claim 4, further comprising displaying the compressed text on a wearable device.

6. The method of any of claim 1 to claim 5, further comprising translating text associated with the text packet from a first language to a second language.

7. The method of any of claim 1 to claim 6, further comprising translating compressed text from a first language to a second language.

8. The method of any of claim 1 to claim 7, wherein generating the text packet includes:

receiving a stream of audio representing speech;

generating text based on the stream of audio;

generating a first text packet including a first portion of the generated text; and

generating a second text packet including the first portion and a second portion of the generated text.

9. The method of any of claim 1 to claim 8, further comprising:

detecting a pause in the speech; and

in response to detecting the pause in the speech, ending the generating of the compressed text representing the speech.

10. The method of claim 9, wherein the pause in the speech is detected based on a threshold amount of time without speech.

11. A non-transitory computer-readable storage medium comprising instructions stored thereon that, when executed by at least one processor, are configured to cause a computing system to:

generate a text packet based on speech;

store the text packet in a first queue;

move the text packet to a second queue based on a criteria; and

generate compressed text representing the speech based on the text packet in the second queue.

12. The non-transitory computer-readable storage medium of claim 11, wherein

the second queue includes a plurality of text packets, and

the generating of the compressed text representing the speech is based on the plurality of text packets.

13. The non-transitory computer-readable storage medium of claim 11 or 12, wherein the instructions are further configured to cause the computing system to:

identify a quantity of stagnated text; and

in response to determining that the quantity of stagnated text satisfies a criteria, restart the generating of the compressed text.

14. The non-transitory computer-readable storage medium of any of claim 11 to claim 13, wherein the text packet includes a predefined number of words.

15. The non-transitory computer-readable storage medium of any of claim 11 to claim 14, wherein the instructions are further configured to cause the computing system to display the compressed text on a wearable device.

16. The non-transitory computer-readable storage medium of any of claim 11 to claim 15, wherein the instructions are further configured to cause the computing system to translate text associated with the text packet from a first language to a second language.

17. The non-transitory computer-readable storage medium of any of claim 11 to claim 16, wherein the instructions are further configured to cause the computing system to translate compressed text from a first language to a second language.

18. The non-transitory computer-readable storage medium of any of claim 11 to claim 17, wherein generating the text packet includes:

receiving a stream of audio representing speech;

generating text based on the stream of audio;

generating a first text packet including a first portion of the generated text; and

generating a second text packet including the first portion and a second portion of the generated text.

19. The non-transitory computer-readable storage medium of any of claim 11 to claim 18, wherein the instructions are further configured to cause the computing system to:

detect a pause in the speech; and

in response to detecting the pause in the speech, end the generating of the compressed text representing the speech.

20. The non-transitory computer-readable storage medium of claim 19, wherein the pause in the speech is detected based on a threshold amount of time without speech.

21. An apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:

generate a text packet based on speech;

store the text packet in a first queue;

move the text packet to a second queue based on a criteria; and

generate compressed text representing the speech based on the text packet in the second queue.

22. The apparatus of claim 21, wherein

the second queue includes a plurality of text packets, and

the generating of the compressed text representing the speech is based on the plurality of text packets.

23. The apparatus of claim 21 or claim 22, wherein the computer program code is further configured to cause the apparatus to:

identify a quantity of stagnated text; and

in response to determining that the quantity of stagnated text satisfies a criteria, restart the generating of the compressed text.

24. The apparatus of any of claim 21 to claim 23, wherein the text packet includes a predefined number of words.

25. The apparatus of any of claim 21 to claim 24, wherein the computer program code is further configured to cause the apparatus to display the compressed text on a wearable device.

26. The apparatus of any of claim 21 to claim 25, wherein the computer program code is further configured to cause the apparatus to translate text associated with the text packet from a first language to a second language.

27. The apparatus of any of claim 21 to claim 26, wherein the computer program code is further configured to cause the apparatus to translate compressed text from a first language to a second language.

28. The apparatus of any of claim 21 to claim 27, wherein generating the text packet includes:

receiving a stream of audio representing speech;

generating text based on the stream of audio;

generating a first text packet including a first portion of the generated text; and

generating a second text packet including the first portion and a second portion of the generated text.

29. The apparatus of any of claim 21 to claim 28, wherein the computer program code is further configured to cause the apparatus to:

detect a pause in the speech; and

in response to detecting the pause in the speech, end the generating of the compressed text representing the speech.

30. The apparatus of claim 29, wherein the pause in the speech is detected based on a threshold amount of time without speech.

Input 105
Compression 110
115-1
The magic that we have here right now is combinin all investments in knowled e ...
<no summary>
115-2
The magic that we have here right now is combining all investments in knowledge into a new computing platform that ultimately lets us build helpful au mented realit experiences...
Combining investments lets us build helpful augmented reality experiences
115-3
The magic that we have here right now is combining all investments in knowledge into a new computing platform that ultimately lets us build helpful augmented reality experiences that start to break down communication barriers ...
Combining investments lets us build helpful augmented reality experiences that break down communication barriers
115-4
The magic that we have here right now is combining all investments in knowledge into a new computing platform that ultimately lets us build helpful augmented reality experiences that start to break down communication barriers and connect as people no matter what language you speak or what community you are a part of.
Combining investments lets us build helpful augmented reality experiences that break down communication barriers and connect people.
120
Combining investments lets us build helpful augmented reality experiences that break down communication barriers and connect people

FIG. 1A

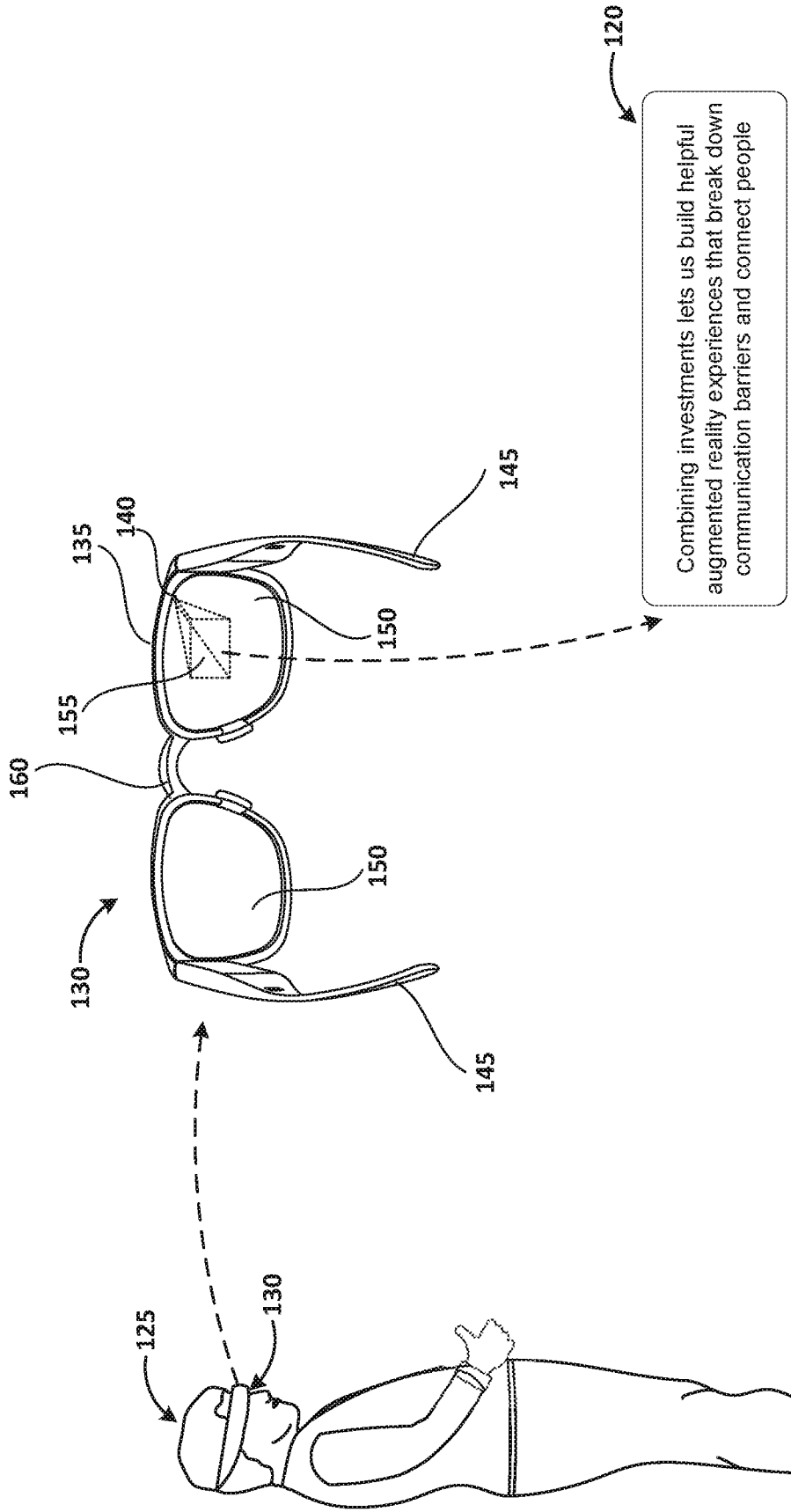

120
Combining investments lets us build helpful augmented reality experiences that break down communication barriers and connect people
145
140
135
150
155
160
150
130
145
130
125

FIG. 1B

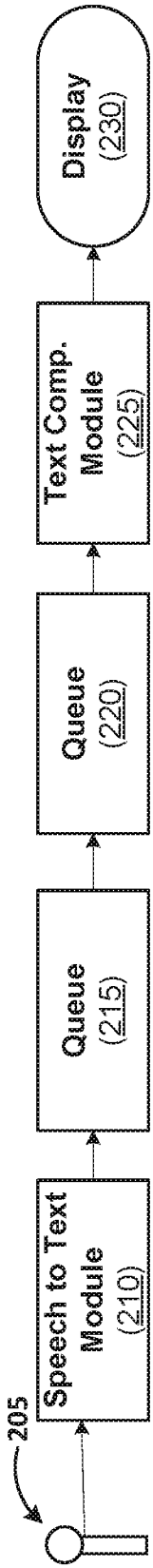


FIG. 2

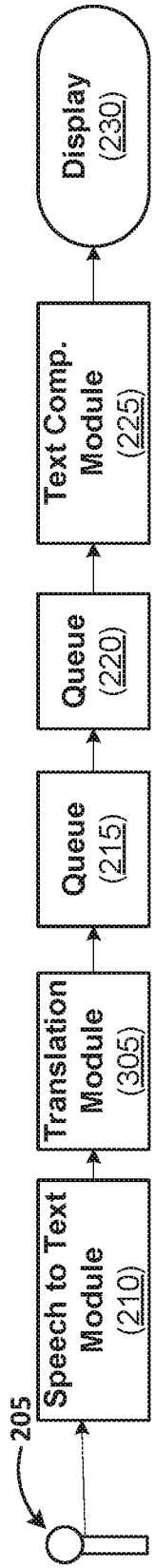


FIG. 3

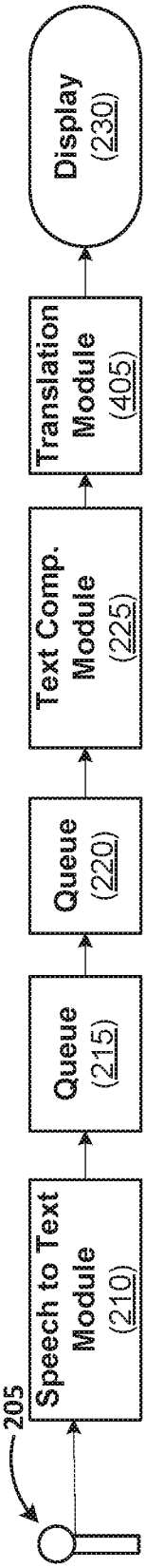


FIG. 4

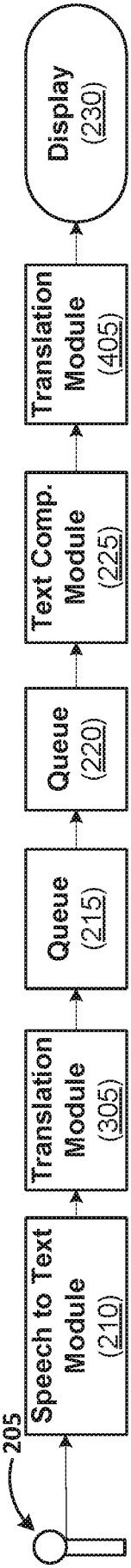


FIG. 5

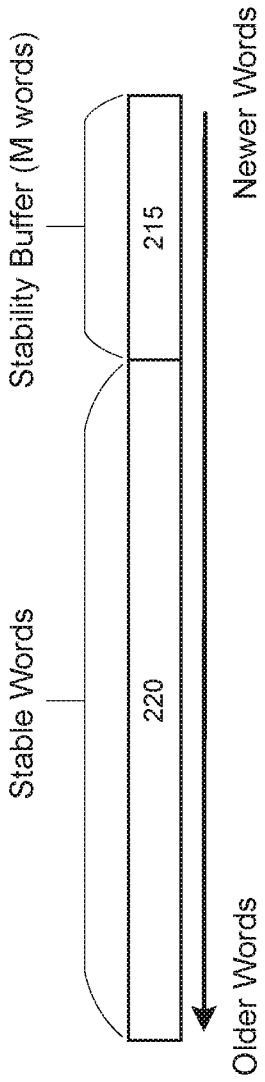

Stability Buffer (M words)
Newer Words
215
Stable Words
220
Older Words

FIG. 6

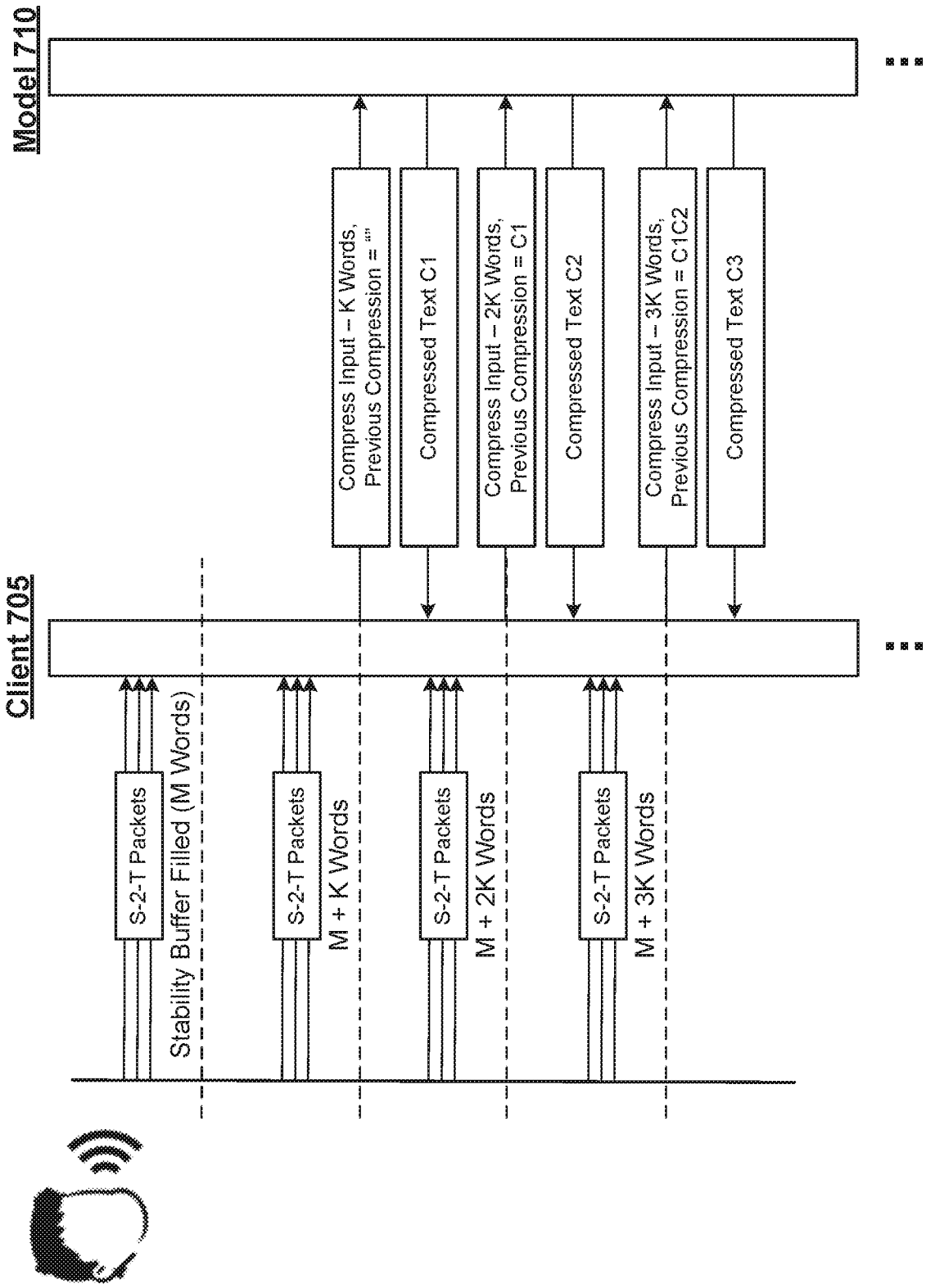

Model 710
Client 705
Compress Input – K Words, Previous Compression = ""
Compressed Text C1
Compress Input – 2K Words, Previous Compression = C1
Compressed Text C2
Compress Input – 3K Words, Previous Compression = C1C2
Compressed Text C3
S-2-T Packets
Stability Buffer Filled (M Words)
S-2-T Packets
M + K Words
S-2-T Packets
M + 2K Words
S-2-T Packets
M + 3K Words

FIG. 7

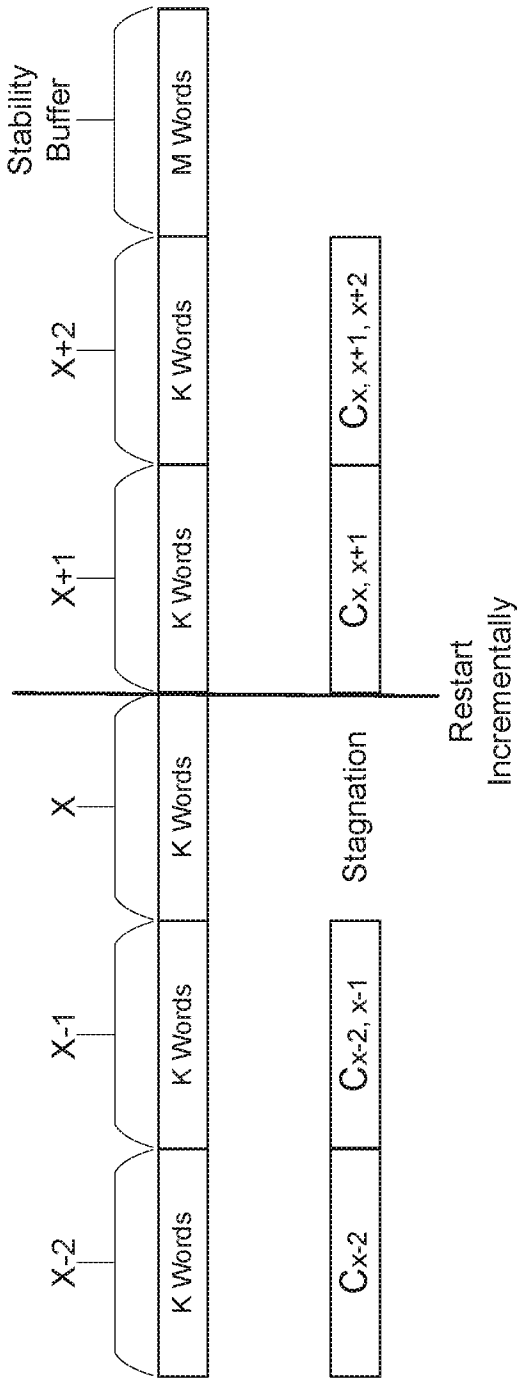

X-2
X-1
X
X+1
X+2
Stability Buffer
K Words
K Words
K Words
K Words
K Words
M Words
Cx-2
Cx-2, x-1
Stagnation
Cx, x+1
Cx, x+1, x+2
Restart Incrementally

FIG. 8

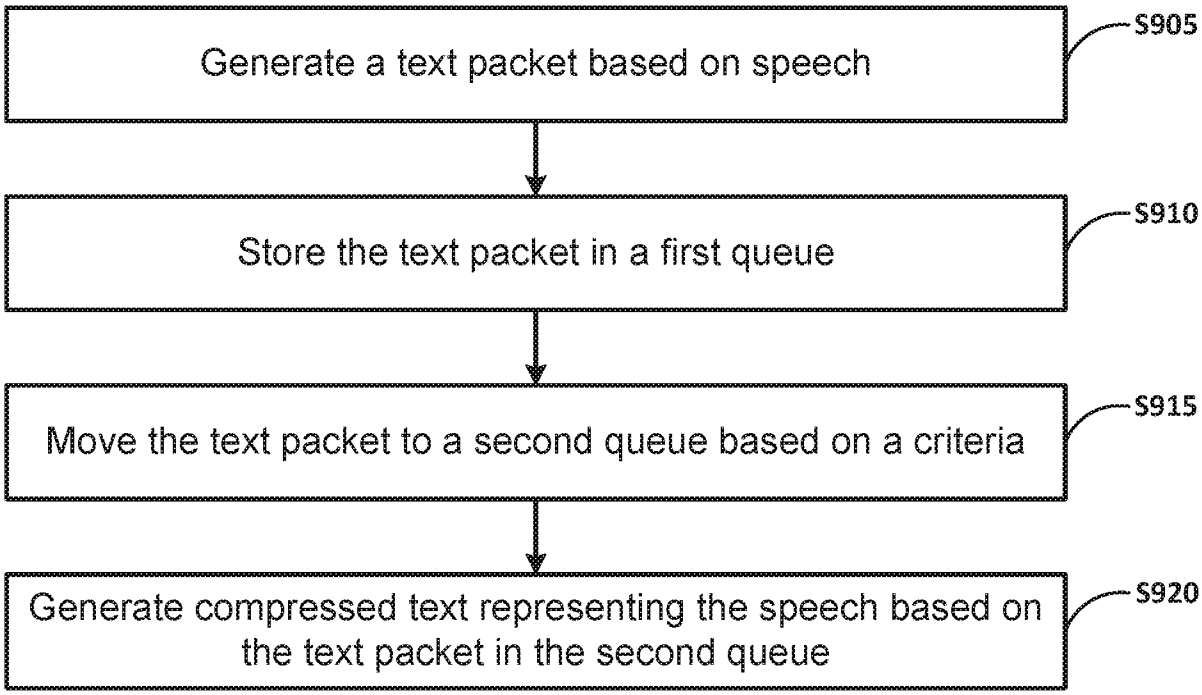

Generate a text packet based on speech
S905
Store the text packet in a first queue
S910
Move the text packet to a second queue based on a criteria
S915
Generate compressed text representing the speech based on the text packet in the second queue
S920

FIG. 9

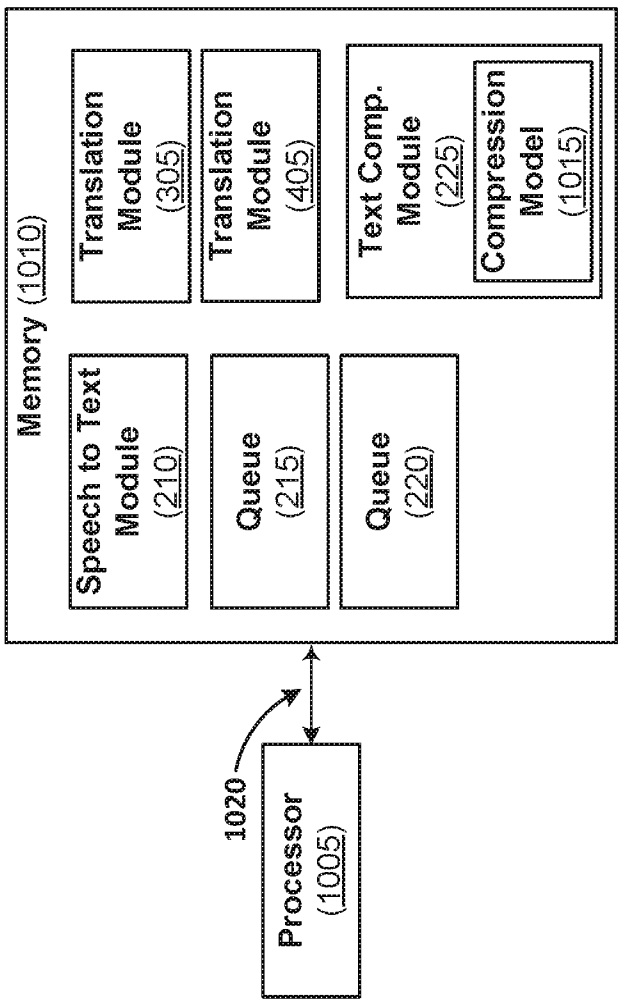

Memory (1010)
Speech to Text Module (210)
Queue (215)
Queue (220)
Translation Module (305)
Translation Module (405)
Text Comp. Module (225)
Compression Model (1015)
1020
Processor (1005)

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2025/010231

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F16/34 G06F16/683 G10L15/26
ADD. G06F3/01 G06F40/58

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F G10L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2023/220201 A1 (GOOGLE LLC [US]) 16 November 2023 (2023-11-16) figure 1 paragraphs [0064] - [0066], [0078], [0085], [0094], [0103], [0193] - [0195]	1-30
A	US 2020/357408 A1 (BOEKWEG SCOTT [US] ET AL) 12 November 2020 (2020-11-12) paragraph [0076]	1-30

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 March 2025	27/03/2025

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Ramos Sánchez, U

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2025/010231

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2023220201 A1	16-11-2023	CN 119096246 A	06-12-2024
		CN 119213441 A	27-12-2024
		EP 4494033 A1	22-01-2025
		EP 4523130 A1	19-03-2025
		US 2023367960 A1	16-11-2023
		WO 2023220198 A1	16-11-2023
		WO 2023220199 A1	16-11-2023
		WO 2023220201 A1	16-11-2023
		WO 2023220202 A1	16-11-2023
		WO 2023220204 A1	16-11-2023
US 2020357408 A1	12-11-2020	US 2020357408 A1	12-11-2020
		US 2022076681 A1	10-03-2022