Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

(11)(21) **3 062 891**

(12) **BREVET CANADIEN**
CANADIAN PATENT

(13) **C**

(86) **Date de dépôt PCT/PCT Filing Date:** 2018/05/18

(87) **Date publication PCT/PCT Publication Date:** 2018/11/22

(45) **Date de délivrance/Issue Date:** 2023/01/03

(85) **Entrée phase nationale/National Entry:** 2019/11/07

(86) **N° demande PCT/PCT Application No.:** US 2018/033487

(87) **N° publication PCT/PCT Publication No.:** 2018/213763

(30) **Priorités/Priorities:** 2017/05/19 (US62/508,977);
2017/07/25 (US62/536,959); 2018/05/17 (US15/982,841)

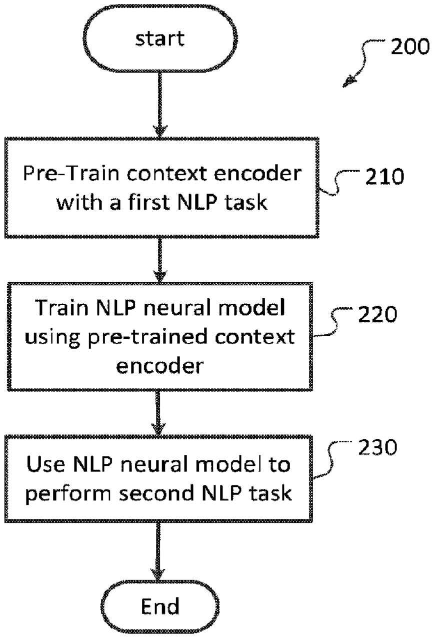
(51) **Cl.Int./Int.Cl.** ***G06F 40/40*** (2020.01),
G06F 40/12 (2020.01), ***G06F 40/279*** (2020.01),
G06F 40/30 (2020.01), ***G06F 40/58*** (2020.01),
G06N 3/02 (2006.01), ***G06N 3/08*** (2006.01)

(72) **Inventeurs/Inventors:**
MCCANN, BRYAN, US;
XIONG, CAIMING, US;
SOCHER, RICHARD, US

(73) **Propriétaire/Owner:**
SALESFORCE.COM, INC., US

(74) **Agent:** SMART & BIGGAR LP

(54) **Titre : TRAITEMENT DE LANGAGE NATUREL A L'AIDE DE VECTEURS DE MOTS SPECIFIQUES AU CONTEXTE**
(54) **Title: NATURAL LANGUAGE PROCESSING USING CONTEXT-SPECIFIC WORD VECTORS**


start
200
Pre-Train context encoder with a first NLP task
210
Train NLP neural model using pre-trained context encoder
220
Use NLP neural model to perform second NLP task
230
End

(57) **Abrégé/Abstract:**
A system is provided for natural language processing. In some embodiments, the system includes an encoder for generating context-specific word vectors for at least one input sequence of words. The encoder is pre-trained using training data for performing a first natural language processing task. A neural network performs a second natural language processing task on the at least one input sequence of words using the context-specific word vectors. The first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder. In some embodiments, the first natural processing task can be machine translation, and the second natural processing task can be one of sentiment analysis, question classification, entailment classification, and question answering.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2018 (22.11.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/213763 A1

(51) International Patent Classification:
G06N 3/04 (2006.01) *G06F 17/28* (2006.01)

(21) International Application Number:
PCT/US2018/033487

(22) International Filing Date:
18 May 2018 (18.05.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/508,977	19 May 2017 (19.05.2017)	US
62/536,959	25 July 2017 (25.07.2017)	US
15/982,841	17 May 2018 (17.05.2018)	US

(71) Applicant: SALESFORCE.COM, INC. [US/US]; The Landmark @ One, Market Street, Suite 300, San Francisco, California 94105 (US).

(72) Inventors: MCCANN, Bryan; The Landmark @ One Market, Suite 300, San Francisco, CA 94105 (US). XIONG, Caiming; The Landmark @ One Market, Suite 300, San Francisco, CA 94105 (US). SOCHER, Richard; The Landmark @ One Market, Suite 300, San Francisco, CA 94105 (US).

(74) Agent: WOO, Philip et al.; Haynes & Boone, LLP, IP Section, 2323 Victory Avenue, Suite 700, Dallas, Texas 75219 (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, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) Title: NATURAL LANGUAGE PROCESSING USING CONTEXT-SPECIFIC WORD VECTORS

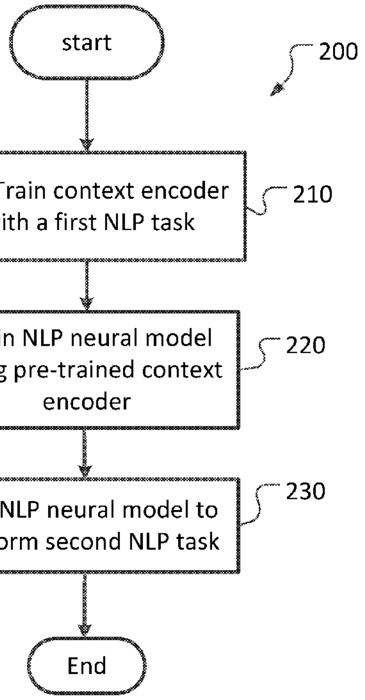

start
200
Pre-Train context encoder with a first NLP task
210
Train NLP neural model using pre-trained context encoder
220
Use NLP neural model to perform second NLP task
230
End

FIG. 2

(57) Abstract: A system is provided for natural language processing. In some embodiments, the system includes an encoder for generating context-specific word vectors for at least one input sequence of words. The encoder is pre-trained using training data for performing a first natural language processing task. A neural network performs a second natural language processing task on the at least one input sequence of words using the context-specific word vectors. The first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder. In some embodiments, the first natural processing task can be machine translation, and the second natural processing task can be one of sentiment analysis, question classification, entailment classification, and question answering.

[Continued on next page]

SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, 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, 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 the identity of the inventor (Rule 4.17(i))*

— *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))*

NATURAL LANGUAGE PROCESSING USING CONTEXT-SPECIFIC WORD VECTORS

[0001]

TECHNICAL FIELD

[0002] The present disclosure relates generally to neural networks and more specifically to neural networks for natural language processing using context-specific word vectors.

BACKGROUND

[0003] Neural networks have demonstrated great promise as a technique for automatically analyzing real-world information with human-like accuracy. In general, neural network models receive input information and make predictions based on the input information. For example, a neural network classifier may predict a class of the input information among a predetermined set of classes. Whereas other approaches to analyzing real-world information may involve hard-coded processes, statistical analysis, and/or the like, neural networks learn to make predictions gradually, by a process of trial and error, using a machine learning process. A given neural network model may be trained using a large number of training examples, proceeding iteratively until the neural network model begins to consistently make similar inferences from the training examples that a human might make. Neural network models have been shown to outperform and/or have the potential to outperform other computing techniques in a number of applications. Indeed, some applications have even been identified in which neural networking models exceed human-level performance.

Date Reçue/Date Received 2021-04-07

SUMMARY

[0003a] Accordingly, there is described a system for natural language processing, the system comprising: a multi-layer neural network; wherein the system is configured to: convert at least one input sequence of words in a first language to a sequence of word vectors; generate context-specific word vectors for the at least one input sequence of words in the first language using an encoder pre-trained using training data for translating phrases from the first language to phrases of a second language; concatenate the word vectors and the context-specific word vectors; and perform a first natural language processing task on the least one input sequence of words in the first language using the concatenated word vectors and context-specific word vectors.

[0003b] There is also described a system for natural language processing, the system comprising: an encoder for generating context-specific word vectors for at least one input sequence of words, wherein the encoder is pre-trained using training data for performing a first natural language processing task; and a neural network for performing a second natural language processing task on the at least one input sequence of words using the context-specific word vectors, wherein the first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder.

[0003c] There is also described a method comprising: using an encoder, generating context-specific word vectors for at least one input sequence of words, wherein the encoder is pre-trained using training data for performing a first natural language processing task; and using a neural network, performing a second natural language processing task on the at least one input sequence of words using the context-specific word vectors, wherein the first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Figure 1 is a simplified diagram of a computing device according to some embodiments.

[0005] Figure 2 is a simplified diagram of a method for pre-training an encoder on a first natural language processing (NLP) task and performing a second NLP task using same according to some embodiments.

[0006] Figure 3 is a simplified diagram illustrating the pre-training of an encoder according to some embodiments.

[0007] Figure 4 illustrates an example of word vectors for a sequence of words.

[0008] Figure 5 is a simplified diagram illustrating the pre-training of an encoder on an NLP task of translation according to some embodiments

[0009] Figure 6 is a simplified diagram of a method for pre-training an encoder on an NLP task of translation according to some embodiments.

[0010] Figure 7 is a simplified diagram of a system for natural language processing according to some embodiments.

[0011] Figure 8 is a simplified diagram of a system for natural language processing using an encoder pre-trained on an NLP task of translation according to some embodiments.

[0012] Figures 9 and 10 are simplified diagrams comparing performance of systems for natural language processing based on different input encodings.

[0013] Figure 11 is a table illustrating performance results of systems for natural language processing based on different input representations.

[0014] In the figures, elements having the same designations have the same or similar functions.

DETAILED DESCRIPTION

[0015] This description and the accompanying drawings that illustrate aspects, embodiments, implementations, or applications should not be taken as limiting—the claims define the protected invention. Various mechanical, compositional, structural, electrical, and operational changes may be made without departing from the spirit and scope of this description and the claims. In some instances, well-known circuits, structures, or techniques have not been shown or described in detail as these are known to one skilled in the art. Like numbers in two or more figures represent the same or similar elements.

[0016] In this description, specific details are set forth describing some embodiments consistent with the present disclosure. Numerous specific details are set forth in order to provide a thorough understanding of the embodiments. It will be apparent, however, to one skilled in the art that some embodiments may be practiced without some or all of these specific details. The specific embodiments disclosed herein are meant to be illustrative but not limiting. One skilled in the art may realize other elements that, although not specifically described here, are within the scope and the spirit of this disclosure. In addition, to avoid unnecessary repetition, one or more features shown and described in association with one embodiment may be incorporated into other embodiments unless specifically described otherwise or if the one or more features would make an embodiment non-functional.

[0017] Natural language processing (NLP) is one class of problems to which neural networks may be applied. NLP can be used to instill new neural networks with an understanding of individual words and phrases. For most problems or tasks in NLP, however, understanding context is also important. Translation models need to understand, for example, how the words in an English sentence work together in order to generate a German translation. Likewise, summarization models need context in order to know which words are most important. Models performing sentiment analysis need to understand how to pick up on key words that change the sentiment expressed by others. And question answering models rely on an understanding of how words in a question shift the importance of words in a document. Accordingly, it is desirable to develop a way to initialize neural networks for NLP with an understanding of how various words might relate to other words or how context influences a word’s meaning.

[0018] According to some embodiments, a neural network is taught how to understand words in context by training it on a first NLP task—e.g., teaching it how to translate from English to German. The trained network can then be reused in a new or other neural network that performs a second NLP task—e.g., classification, question answering, sentiment analysis, entailment classification, language translation, etc. The pre-trained network's outputs—context-specific word vectors (CoVe)—are provided as inputs to new networks that learn other NLP tasks. Experiments show that providing CoVe to these new networks can improve their performance, thus validating that various NLP models or tasks can benefit from using a neural network that has already learned how to contextualize words.

[0019] In some embodiments, various NLP models or tasks—such as classification, question answering, sentiment analysis, and translation—can be improved by using context-specific word vectors generated by training an encoder with a NLP task that may be different from the NLP task to be performed. More generally speaking, significant gains have been made through transfer and multi-task learning between synergistic tasks. In many cases, these synergies can be exploited by architectures that rely on similar components. Embodiments disclosed herein use networks that have already learned how or been trained to contextualize words to give other neural networks an advantage in learning to understand other parts of natural language.

[0020] Figure 1 is a simplified diagram of a computing device 100 according to some embodiments. As shown in Figure 1, computing device 100 includes a processor 110 coupled to memory 120. Operation of computing device 100 is controlled by processor 110. And although computing device 100 is shown with only one processor 110, it is understood that processor 110 may be representative of one or more central processing units, multi-core processors, microprocessors, microcontrollers, digital signal processors, field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), graphics processing units (GPUs), tensor processing units (TPUs), and/or the like in computing device 100. Computing device 100 may be implemented as a stand-alone subsystem, as a board added to a computing device, and/or as a virtual machine.

[0021] Memory 120 may be used to store software executed by computing device 100 and/or one or more data structures used during operation of computing device 100. Memory 120 may include one or more types of machine readable media. Some common forms of machine readable media may include floppy disk, flexible disk, hard disk, magnetic tape, any other magnetic medium, CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, RAM, PROM, EPROM, FLASH-EPROM, any other memory chip or cartridge, and/or any other medium from which a processor or computer is adapted to read.

[0022] Processor 110 and/or memory 120 may be arranged in any suitable physical arrangement. In some embodiments, processor 110 and/or memory 120 may be implemented on a same board, in a same package (e.g., system-in-package), on a same chip (e.g., system-on-chip), and/or the like. In some embodiments, processor 110 and/or memory 120 may include distributed, virtualized, and/or containerized computing resources. Consistent with such embodiments, processor 110 and/or memory 120 may be located in one or more data centers and/or cloud computing facilities. In some examples, memory 120 may include non-transitory, tangible, machine readable media that includes executable code that when run by one or more processors (e.g., processor 110) may cause the one or more processors to perform any of the methods described further herein.

[0023] As shown, memory 120 includes a neural network 130. Neural network 130 may be used to implement and/or emulate any of the neural networks described further herein. In some examples, neural network 130 may include a multi-layer or deep neural network. According to some embodiments, examples of multi-layer neural networks include the ResNet-32, DenseNet, PyramidNet, SENet, AWD-LSTM, AWD-QRNN and/or the like neural networks. The ResNet-32 neural network is described in further detail in He, et al., “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385,* submitted on December 10, 2015; the DenseNet neural network is described in further detail in Iandola, et al., “Densenet: Implementing Efficient Convnet Descriptor Pyramids,” *arXiv:1404.1869,* submitted April 7, 2014, the PyramidNet neural network is described in further detail in Han, et al., “Deep Pyramidal Residual Networks,” *arXiv:1610.02915,* submitted October 10, 2016; the SENet neural network is described in further detail in Hu, et al., “Squeeze-and-Excitation Networks,” *arXiv:1709.01507,* September 5, 2017; the AWD-LSTM neural network is described in further detail in Bradbury, et al., “Quasi-

Recurrent Neural Networks," *arXiv:1611.01576,* submitted on November 5, 2016.

[0024] According to some embodiments, the neural network 130 may use an encoder that is pre-trained for a first kind of NLP task, such as, for example, translation. The computing device 100 may receive training data that includes one or more sequences of words in a first language (e.g., English), and one or more corresponding sequences of words in a second language (e.g., German) that represent the expected and/or desired translation of the respective first language word sequences. To illustrate, suppose an input word sequence provided to a computing device 100 includes the English word sequence "Let's go for a walk." The corresponding German word sequence is "Lass uns spazieren gehen." Computing device 100 uses this training data to generate and output context-specific word vectors or "context vectors" (CoVe) for the words or sequences of words in the first language. Stated differently, the encoder is taught how to understand words in context by first teaching it how to translate from one language into another (e.g., English to German). Once trained, the encoder may be used by the neural network 130 to perform a second kind of NLP task—e.g., sentiment analysis (Stanford Sentiment Treebank (SST), IMDb), question classification (TREC), entailment classification (Stanford Natural Language Inference Corpus (SNLI)), question answering (Stanford Question Answering Dataset (SQuAD)) and/or the like. To this end, the computing device 100 receives input 150 for the second kind of NLP task, and generates results 160 for that task.

[0025] Figure 2 is a simplified diagram of a method 200 for pre-training an encoder on a first NLP task and performing a second NLP task using the same, according to some embodiments. One or more of the processes 210-220 of method 200 may be implemented, at least in part, in the form of executable code stored on non-transitory, tangible, machine-readable media that when run by one or more processors may cause the one or more processors to perform one or more of the processes 210-230. In some embodiments, method 200 can be performed by computing device 100 of Figure 1.

[0026] According to some embodiments, method 200 utilizes transfer learning, or domain adaptation. Transfer learning has been applied in a variety of areas where researchers identify synergistic relationships between independently collected datasets. In some embodiments, the source domain of transfer learning is machine translation.

[0027] At a process 210, an encoder of a neural network is pre-trained using training data for performing the first NLP task. In some embodiments, the first NLP task can be translation. The nature of the translation task has appealing properties for training a general context encoder—e.g. translation seems to require a more general sense of language understanding than other NLP tasks, like text classification. During training, the encoder is provided with training and/or testing data 150 that, in some embodiments, may include one or more sequences of words in a first language (e.g., English), and one or more corresponding sequences of words in a second language (e.g., German). The training data 150 can be one or more machine translation (MT) datasets. Machine translation is a suitable source domain for transfer learning because the task, by nature, requires the model to faithfully reproduce a sentence in the target language without losing information in the source language sentence. Moreover, there is an abundance of machine translation data that can be used for transfer learning; indeed, machine translation training sets are much larger than those for most other NLP tasks. Possible training sets include various English-German machine translation (MT) datasets. For example, the WMT 2016 multi-modal translation shared task—often referred to as "Multi30k" and described in further detail in Specia, et al., "A shared task on multimodal machine translation and crosslingual image description," *Proceedings of the 1st Conference on Machine Translation*, WMT, 2016, pp. 543- 553 is a dataset, consisting of 30,000 sentence pairs that briefly describe Flickr captions. Due to the nature of image captions, this dataset contains sentences that are, on average, shorter and simpler than those from larger counterparts. The 2016 version of the machine translation task prepared for the International Workshop on Spoken Language Translation—described in further detail in Cettolo, et al., "The IWSLT 2015 evaluation campaign," *In International Workshop on Spoken Language Translation*, 2015, is a larger dataset, consisting of 209,772 sentence pairs from transcribed TED presentations that cover a wide variety of topics with more conversational language than in other machine translation datasets. The news translation shared task from WMT 2017 is a large MT dataset, consisting of roughly 7 million sentence pairs that comes from web crawl data, a news and commentary corpus, European Parliament proceedings, and European Union press releases. These three MT datasets may be referred to as MT-Small, MT-Medium, and MT-Large, respectively. Each of these MT datasets is tokenized using the Moses Toolkit , which is described in further detail in Koehn, et al., "Moses: Open source toolkit for statistical machine translation," *Proceedings of the 45th Annual Meeting of the Association of the Computational Linguistics*, 2007, pp. 177-180.

[0028] The encoder generates or outputs context vectors (or CoVe) 160 for the words or sequences in the first language. The context vectors from encoders trained on MT-Small, MT-Medium, and MT-Large may be referred to as CoVe-S, CoVe-M, and CoVe-L. The pre-trained encoder of the neural network can then be reused or applied to one or more other NLP tasks.

[0029] At a process 220, a new or another neural network 130 is trained for second NLP task using the pre-trained context encoder. The pre-trained encoder's outputs—context vectors (or CoVe) for the words or sequences in the first language—are provided as inputs to the new or other neural network 130 that learns or executes other NLP tasks performed on the same language, such as classification, question answering, sentiment analysis, other machine translation tasks, and/or the like. In some embodiments, fixed-length representations obtained from neural machine translation (NMT) encoders are transferred in for the training. In some embodiments, representations for each token in an input sequence are transferred in for training. The latter approach makes the transfer of the pre-trained context encoder for the other NLP task more directly compatible with subsequent long-term short-term memories (LSTMs), attention mechanisms, and, in general, layers that expect input sequences. This additionally facilitates the transfer of sequential dependencies between encoder states. In some embodiments, the pre-trained encoder is not further trained during process 220.

[0030] At a process 230, the neural network 130 is used to perform the second NLP task. The computing device 100 receives input 150 for the second NLP task, and generates results 160 for that task. Experiments show that providing the neural network 130 with context vectors from an encoder pre-trained on a first NLP task (e.g., machine translation)can improve its performance for a second NLP task (e.g., classification, question answering, sentiment analysis).

[0031] Aspects or embodiments for each of these processes 210-230 of method 200 are described in more detail herein.

[0032] Figure 3 is a simplified diagram illustrating the pre-training of an encoder 310 according to some embodiments. In some embodiments, the encoder 310 may include or be implemented with one or more long-term short-term memory (LSTM) encoders.

[0033] The encoder 310 receives training data, which may be in the form of word vectors 320 for one or more sequences of words in a first language (e.g., English). Instead of reading sequences of words as text, deep learning models read sequences of word vectors. A word vector associates each word in the language with a list of numbers. Many deep learning models for NLP rely on word vectors to represent the meaning of individual words.

[0034] Figure 4 illustrates an example of word vectors for a sequence of words: "Let's go for a walk." In some embodiments, the word vectors 320 of a model are initialized to lists of random numbers before the model is trained for a specific task. In some embodiments, the word vectors 320 of a model can be initialized with those obtained by running methods like word2vec, GloVe, or FastText. Each of those methods defines a way of learning word vectors with useful properties. The first two methods work off of the hypothesis that at least part of a word's meaning is tied to how it is used. word2vec trains a model to take in a word and predict a local context window; the model sees a word and tries to predict the words around it. GloVe takes a similar approach, but it also explicitly adds statistics about how often each word occurs with each other word. In both cases, each word is represented by a corresponding word vector, and training forces the word vectors to correlate with each other in ways that are tied to the usage of the word in natural language. With reference to the specific example of "Let's go for a walk" shown in Figure 4, algorithms like word2vec and GloVe produce word vectors correlated with the word vectors that regularly occur around it in natural language. In this way the vector for "go" comes to mean that the word "go" appears around words like "Let's," "for," "a," and "walk."

[0035] Referring back to Figure 3, the encoder 310 is trained by having it perform a first NLP task which, in some embodiments, can be machine translation (MT) of the word sequence in a first language (e.g., "Let's go for a walk") into a corresponding word sequence in a second language (e.g., "Lass uns spazieren gehen"). To accomplish this training, the encoder 310 interacts with a decoder 330 to generate the translation 340. In some embodiments, the LSTM encoders are trained on several machine translation datasets. Experiments show that the quantity of training data used to train the MT-LSTM is positively correlated with performance on downstream tasks, such as when the encoder is used or employed for a second NLP task. This is yet another advantage of using MT as a training task, as data for MT is more abundant than for most other supervised NLP tasks, and it suggests that higher quality MT-LSTMs carry over more useful information. While machine translation might seem unrelated to other NLP tasks, such as text classification and question answering, this reinforces the idea that machine translation is a good candidate NLP task for models with a stronger sense of natural language understanding.

[0036] While Figure 3 is a high-level diagram, Figure 5 illustrates more details for the pre-training of the encoder 310 on the NLP task of machine translation according to some embodiments. And Figure 6 shows a corresponding method 600 for pre-training the encoder illustrated in Figure 5.

[0037] With reference to Figures 5 and 6, the method 600 starts with a process 602. At process 602, word vectors 320a-e for a sequence of words in a first or source language $w^x = [w^x{}_1, \ldots, w^x{}_n]$ (e.g., English - “Let’s go for a walk”) are input or provided to the encoder 310. And word vectors 540 for a sequence of words in a second or target language $w^z = [w^z{}_1, \ldots, w^z{}_n]$ (e.g., German – “Lass uns spazieren gehen”) are input or provided to the decoder 330. Let GloVe(w^x) be a sequence of GloVe vectors corresponding to the words in w^x, and let z be a sequence of randomly initialized word vectors corresponding to the words in w^z.

[0038] In some embodiments, the encoder 310 includes or is implemented with a recurrent neural network (RNN). RNNs are deep learning models that process vector sequences of variable length. This makes RNNs suitable for processing sequences of word vectors 320a-e. In some embodiments, the encoder 310 can be implemented with one or more long-term short-term memory (LSTM) encoders 510a-e, which are a specific kind of RNN capable of handling long word sequences.

[0039] At a process 604, the encoder processes the sequence of word vectors 320a-e to generate one or more new vector 520a-e, each called a hidden vector. In some embodiments, the encoder 310 encodes the input sequence, for example, with each LSTM 510a-e taking in a respective word vector 320a-e and outputting the respective hidden vector 520a-e. The encoder

310 is run forward so that information generated by an LSTM encoder 510 operating on a word vector 320 appearing earlier in the input sequence is passed to LSTM encoders 510 operating on word vectors 320 appearing later in the sequence. This allows the hidden vectors of the later LSTM encoders 510 to incorporate information for the earlier word vectors 320. In some embodiments, the encoder 310 is also run backwards so that the LSTM encoders 510a-e can generate or output hidden vectors that incorporate information from words that appear later in the sequence. These backwards output vectors can be concatenated with the forwards output vectors to yield a more useful hidden vector. Each pair of forward and backward LSTMs can be treated as a unit, and is typically referred to as a bidirectional LSTM. A bidirectional LSTM encoder incorporates information that precedes and follows the respective word. The LSTM trained on machine translation may be referred to as MT-LSTM. The first bidirectional LSTM 510a processes its entire sequence before passing outputs to the second LSTM 510b; the second bidirectional LSTM 510b does the same, and so on. Each of the bidirectional LSTMs (or biLSTM) generates an output at each time step i as $\mathrm{h_i}$ as the concatenation of $\mathrm{h_i^{\rightarrow}}$ and $\mathrm{h_i^{\leftarrow}}$ according to: $\mathrm{h_i^{\rightarrow} = LSTM(x_i, h_{i-1}^{\rightarrow}); h_i^{\leftarrow} = LSTM(x_i, h_{i+1}^{\leftarrow})}$, where x is the input to the respective biLSTM and LSTM corresponds to a long-term short-term memory network. The encoder 310 with bidirectional LSTMs 510a-e takes in a sequence of word vectors 320a-e, runs a forward and a backward LSTM operation, concatenates the outputs corresponding to the same input, and returns the resulting sequence of hidden vectors *h* 520a-e for the first language (e.g., English) sentence.

$$h = \text{MT-LSTM}(\text{GloVe}(w^x)). \tag{1}$$

For machine translation, the MT-LSTM supplies the context for an attentional decoder that produces a distribution over output words $p(\widehat{w}_t^z | H, w_1^Z, \ldots, w_{t-1}^Z)$ at each time-step *t*, where *H* refers to the elements of *h* stacked along the time dimension.

[0040] At a process 606, the decoder 330 is initialized with the final states/hidden vectors *h* 520a-e from encoder 310. The decoder 330 includes or is implemented with another neural network that references those hidden vectors *h* 520a-e as it generates or translates into the second or target language (e.g., German) sentence. Like the encoder 310, in some embodiments, the decoder 330 can include or be implemented with one or more LSTMs 530a-b, which can be bidirectional. At time-step t, the decoder 330 first uses the two-layer, unidirectional LSTM to produce a hidden state vector 550 (h_t^{dec}) based on the previous target embedding (z_{t-1}) and a context-adjusted hidden state $\left(\tilde{h}_{t-1}\right)$:

$$h_t^{\text{dec}} = \text{LSTM}\left(\left[z_{t-1}; \tilde{h}_{t-1}\right], h_{t-1}^{\text{dec}}\right). \tag{2}$$

The first of the decoder LSTMs 530a is initialized from the final states h of the encoder 310 and reads in a special German word vector 540a to start.

[0041] At a process 610, a word from the sequence in the first language is selected. In some embodiments, an attention mechanism 560 looks back at the hidden vectors 520a-e in order to decide which word of the first language (e.g., English) sentence to translate next. The attention mechanism 560 computes a vector of attention weights α representing the relevance of each encoding time-step to the current decoder state.

$$\alpha_t = \text{softmax}\left(H\left(W_1 h_t^{\text{dec}} + b_1\right)\right) \tag{3}$$

[0042] At a process 612, the attention mechanism 560 generates a new vector 570, which can be referred to as the context-adjusted state. The attention mechanism 560 uses the weights α as coefficients in an attentional sum that is concatenated with the decoder state and passed through a tanh layer to form the context-adjusted hidden state $\tilde{h}$:

$$\tilde{h}_t = \left[\tanh\left(W_2 H^{\top} \alpha_t + b_2\right); h_t^{\text{dec}}\right] \tag{4}$$

In other words, the attention mechanism 560 uses the decoder state vector 550a to determine how important each hidden vector 520a-e is, and then produces the context-adjusted state 570 to record its observation.

[0043] At a process 614, a generator 580 looks at the context-adjusted state 570 to determine the word in the second language (e.g., German) to output. The context-adjusted state 570 is passed back to the next LSTM 540 so that it has an accurate sense of what it has already translated. The distribution over output words is generated by a final transformation of the context-adjusted hidden state:

$$p(\hat{w}_t^z | X, w_1^z, \ldots, w_{t-1}^z) = \text{softmax}\left(W_{\text{out}}\tilde{h}_t + b_{\text{out}}\right)$$

[0044] At a process 616, a determination is made as to whether the current word in the first language is the final word in the sequence. If not, decoder 330 repeats processes 610-616 until it has completed generating the translated word sequence in the second language.

[0045] In some examples, training of an MT-LSTM of the encoder 310 uses fixed 300-dimensional word vectors, such as the CommonCrawl-840B GloVe model for English word vectors. These word vectors are completely fixed during training, so that the MT-LSTM learns how to use the pretrained vectors for translation. The hidden size of the LSTMs in all MT-LSTMs is 300. Because all MT-LSTMs are bidirectional, they output 600-dimensional vectors. The encoder 310 can be trained with stochastic gradient descent with a learning rate that begins at 1 and decays by half each epoch after the validation perplexity increases for the first time. Dropout with ratio 0:2 may be applied to the inputs and outputs of all layers of the encoder 310 and decoder 330.

[0046] When training is finished, the pre-trained encoders can be used to improve the performance of neural models trained for other tasks in natural language processing (NLP). The LSTMs 510 that were trained as an encoder for machine translation can be extracted, and their learning transferred to downstream NLP tasks (e.g., classification, or question answering). The pre-trained LSTMs, which may be referred to as an MT-LSTM, can be used to output hidden vectors for other sentences or word sequences in the first language. These machine translation hidden vectors, when used as inputs to another NLP model, provide or serve as context-specific word vectors or "context vectors" (CoVe). If *w* is a sequence of words and GloVe(*w*) is the corresponding sequence of word vectors produced by the GloVe model, then

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w)) \tag{5}$$

is the sequence of context vectors produced by the MT-LSTM. Referring back to Figure 5, for example, GloVe(*w*) corresponds to 320a-e, and CoVe(*w*) corresponds to 520a-e. In some embodiments, for the downstream NLP task, for an input sequence *w*, each vector in GloVe(*w*) can be concatenated with its corresponding vector in CoVe(*w*) to yield a vector sequence (*w*):

$$\tilde{w} = [\mathrm{GloVe}(w); \mathrm{CoVe}(w)] \qquad (6)$$

[0047] Some examples of computing devices, such as computing device 100 may include non-transitory, tangible, machine readable media that include executable code that when run by one or more processors (e.g., processor 110) may cause the one or more processors to perform the processes of method 600. Some common forms of machine readable media that may include the processes of method 600 are, for example, floppy disk, flexible disk, hard disk, magnetic tape, any other magnetic medium, CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, RAM, PROM, EPROM, FLASH-EPROM, any other memory chip or cartridge, and/or any other medium from which a processor or computer is adapted to read.

[0048] Figure 7 is a simplified diagram illustrating a system 700 for natural language processing according to some embodiments. System 700 includes one or more encoders 710, pre-trained on a first NLP task, such as, for example machine translation, as described herein, and now re-used as part of a new model. In some embodiments, each encoder 710 is consistent with encoder 310. In some embodiments, each encoder 710 includes or is implemented with one or more pre-trained MT-LSTMs. Pre-trained encoder 710 is capable of providing or generating context vectors (CoVe) from input word vectors 720.

[0049] Word vectors 720 of a model can be initialized with those obtained by running methods like word2vec, FastText, or GloVe, each of which defines a way of learning word vectors with useful properties. In some embodiments, the word vectors 720 of a model are initialized to lists of random numbers before the model is trained for a specific task.

[0050] System 700 also includes neural model 730 for performing a second, specific NLP task, such as, for example, sentiment analysis (Stanford Sentiment Treebank (SST), IMDb), question classification (TREC), entailment classification (Stanford Natural Language Inference Corpus (SNLI)), question answering (Stanford Question Answering Dataset (SQuAD)) and/or the like. In some embodiments, neural model 730 is consistent with neural network of model 130. Neural model 730 is provided with the context vectors (CoVe) from pre-trained encoders 710. In some embodiments, the context vectors (CoVe) from encoder 710 may be appended or concatenated with the word vectors 720 (e.g., GloVe) that are typically used as inputs to these kinds of neural models (see Eq. 6), and the results provided to the neural model 730. This approach improves the performance of the neural model 730 for downstream tasks over that of baseline models using pre-trained word vectors alone. In general, context vectors (CoVe) can be used with any neural model 730 that represents its inputs as a sequence of vectors. Experiments have shown the advantages of using pre-trained MT-LSTMs to generate context vectors (CoVe) for neural models performing NLP tasks such as text classification and question answering models. For the Stanford Sentiment Treebank (SST) and the Stanford Natural Language Inference Corpus (SNLI), the use of context vectors (CoVe) pushes performance of the baseline model to the state of the art.

[0051] Figure 8 is a diagram illustrating a system 800 for natural language processing using one or more encoders 810 pre-trained on an NLP task of translation according to some embodiments. In some embodiments, each encoder 810 is consistent with encoder 310, 710. System 800 may include or be implemented with a multi-layer neural network or neural model 830 for performing a specific NLP task—such as, for example, question classification (TREC), question answering (SQuAD), sentiment analysis (SST, IMDb), entailment classification (SNLI), and/or the like—which is different from the NLP task of translation. In some embodiments, neural model 830 is consistent with neural model 130, 730.

[0052] The neural model 830 of system 800 may be trained for the specific NLP tasks with suitable datasets. For example, training of the neural model 830 for question classification may use the small TREC dataset of open-domain, fact-based questions divided into broad semantic categories, as described in further detail in Voorhees, et al., "The TREC-8 question answering track evaluation," *The Eighth Text Retrieval Conference*, volume 1999, p. 83. This dataset can be the fifty-class or six-class versions of TREC, referred to as TREC-50 and TREC-6, respectively. Both have 4,300 training examples, but TREC-50 has finer-grained labels. For question answering, the neural model 830 can be trained with the Stanford Question Answering Dataset (SQuAD), as described in further detail in Rajpurkar, et al., "SQuAD: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, submitted on June 16, 2016. SQuAD is a large-scale question answering dataset with 87,599 training examples and 10,570 development examples. Examples consist of a paragraph from English Wikipedia and associated question-answer pairs over the paragraph. SQuAD examples assume that the question is answerable and the answer is contained verbatim somewhere in the paragraph. For sentiment analysis, the neural model 830 can be separately trained on two sentiment analysis datasets: the Stanford Sentiment Treebank (SST) (as described in further detail in Socher, et al., "Recursive deep models for semantic compositionality over a sentiment Treebank," *Empirical Methods in Natural Language Processing*, 2013) and the IMDb dataset (as described in further detail in Maas, et al., "Learning word vectors for sentiment analysis," *In Proceedings of the 49th Annual Meetings of the Association for Computational Linguistics: Human Language Technologies*, pp. 142-150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology.P11-1015]). Both of these datasets comprise movie reviews and their sentiment. The binary version of each dataset is used, as well as the five-class version of SST. IMDb contains 22,500 multi-sentence reviews, each of which can be truncated to the first 200 words. SST-2 contains 56,400 examples with the "neutral" class removed and all sub-trees included, and SST-5 contains 94,200 reviews with all classes and sub-trees. For entailment, the neural model 830 can be trained with the Stanford Natural Language Inference Corpus (SNLI), as described in further detail in Bowman, et al., "Recursive neural networks for learning logical semantics," *arXiv preprint arXiv:1406.1827*, submitted on June 6, 2014. SNLI has 550,152 training, 10,000 validation, and 10,000 testing examples. Each example consists of a premise, a hypothesis, and a label specifying whether the premise entails, contradicts, or is neutral with respect to the hypothesis.

[0053] As shown in Figure 8, system 800 includes a neural model 830 for a general biattentive classification network (BCN). This model 830 is designed to handle both single-sequence and two-sequence classification tasks. In the case of single-sequence tasks, the input word sequence is duplicated to form two sequences.

[0054] The two input sequences w^x and w^y are provided as word vectors 820 (e.g., Glove(w)) to system 800 at pre-trained encoders 810. In some embodiments, each encoder 810 is consistent with encoder 310, 710. The encoders 810 are pre-trained on the NLP task of machine translation, and thus provide or generate respective context vectors (CoVe)(w)) from input word vectors 820. In some embodiments, each word vector 820 (e.g., Glove(w)) is concatenated or appended with its corresponding context vectors (CoVe)(w)) to generate sequences of vectors, $\widetilde{w}^x$ and $\widetilde{w}^y$, as described herein (e.g., Eq. 6). The vector sequences, $\widetilde{w}^x$ and $\widetilde{w}^y$ are provided as input to the task-specific portion of the model 830.

[0055] The neural network or model 830 is trained using the pre-trained encoders 810. In some embodiments, the encoders 810 are not further trained when neural network or model 830 is trained.

[0056] The model 830 includes one or more rectifier linear units (ReLUs) 832, which receive the input vector sequences $\widetilde{w}^x$ and $\widetilde{w}^y$. The ReLUs 832 implement or execute a function f that applies a feedforward network with ReLU activation (as described in further detail in Nair et al., “Rectified linear units improve restricted Boltzmann machines,” *Proceedings of the 27th International Conference on Machine Learning*, 2010) to each element of $\widetilde{w}^x$ and $\widetilde{w}^y$. Encoders 834, each of which can include or be implemented as a bidirectional LSTM (biLSTM), process the resulting sequences to obtain task specific representations (x and y):

$$x = \text{biLSTM}\left(f(\tilde{w}^x)\right) \qquad (7)$$

$$y = \text{biLSTM}\left(f(\tilde{w}^y)\right) \qquad (8)$$

These sequences are each stacked along the time axis to generate matrices X and Y.

[0057] In order to compute representations that are interdependent, model 830 uses a biattention mechanism 836, as described in further detail in Seo, et al., “Bidirectional attention flow for machine comprehension,” *International Conference on Learning Representations*, 2017, and Xiong, et al., “Dynamic coattention networks for question answering,” *International Conference on Learning Representations*, 2017. Biattention conditions each representation on the other.

[0058] Using biattention mechanism 836 in neural model 830 provides an advantage, for example, in some NLP classification tasks such as entailment classification and sentiment analysis or classification. Entailment classification involves the processing of two word sequences for which there may be some form of relation—e.g., determining if one sequence being true entails the other sequence, determining if one sequence being true entails the other sequence's negation, or determining if one sequence being true allows the other to be either true or false. An example of sequences for entailment classification could be: (w^x) "two women are discussing circuit," and (w^y) "two people are discussing technology." With this example, sequence w^x entails sequence w^y. Sentiment classification aims to determine the attitude or sentiment of a speaker or author of a word sequence with respect to some topic. Each of these sequences could be provided to a respective channel (e.g., as input for ReLU 832) in the neural model 830. An example of a sequence for entailment classification could be: (w^x) "this movie was a waste of time." This sequence could be repeated and provided to each of the channels in the neural model 830. In some embodiments, the biattention mechanism 836 results in or yields a better outcome for the NLP classification task by combining attention with element-wise features of classification.

[0059] The biattention mechanism 836 first computes an affinity matrix $A = XY^{\mathrm{T}}$. Biattention mechanism 836 then extracts attention weights (A_x and A_y) with column-wise normalization:

$$A_x = \mathrm{softmax}(A) \qquad A_y = \mathrm{softmax}\left(A^{\top}\right) \quad (9)$$

which can be a form of self-attention when the task specific representations are the same ($x = y$). Next, the biattention mechanism 836 uses context summaries (C_x and C_y)

$$C_x = A_x^{\top} X \qquad C_y = A_y^{\top} Y \qquad (10)$$

to condition each sequence on the other.

[0060] Two separate integrators 838 integrate the conditioning information (generated from biattention mechanism 836) into the task specific representations (x and y) for each input sequence. In some embodiments, each integrator 838 which can include or be implemented with a one-layer biLSTM. The biLSTMs operate on the concatenation of the original representations (to ensure no information is lost in conditioning), their differences from the context summaries (C_x and C_y, to explicitly capture the difference from the original signals), and the element-wise products between originals and context summaries (to amplify or dampen the original signals).

$$X_{|y} = \text{biLSTM}\left([X; X - C_y; X \odot C_y]\right) \qquad (11)$$
$$Y_{|x} = \text{biLSTM}\left([Y; Y - C_x; Y \odot C_x]\right) \qquad (12)$$

[0061] Pool mechanisms 840 aggregate the outputs of the bidirectional LSTMs of integrators 838 by pooling along the time dimension. In some embodiments, max and mean pooling can be used to extract features. In some embodiments, adding both min pooling and a parameter-less form of self-attentive pooling has been found to aid in some tasks. Each type of pooling captures a different perspective on the conditioned sequences. The self-attentive pooling computes weights (β_x and β_y) for each time step of the sequence:

$$\beta_x = \text{softmax}\left(X_{|y}v_1 + d_1\right) \qquad \beta_y = \text{softmax}\left(Y_{|x}v_2 + d_2\right) \qquad (13)$$

The weights (β_x and β_y) are used to get weighted summations (x_{self} and y_{self}) of each sequence:

$$x_{\text{self}} = X_{|y}^{\top}\beta_x \qquad y_{\text{self}} = Y_{|x}^{\top}\beta_y \qquad (14)$$

The pooled representations are combined to get one joined representation (x_{pool} and y_{pool}) for all inputs:

$$x_{\text{pool}} = \left[\max(X_{|y}); \text{mean}(X_{|y}); \min(X_{|y}); x_{\text{self}}\right] \qquad (15)$$
$$y_{\text{pool}} = \left[\max(Y_{|x}); \text{mean}(Y_{|x}); \min(Y_{|x}); y_{\text{self}}\right] \qquad (16)$$

[0062] For a NLP task of classification, the joined representation are provided or input into maxout layers 842. The maxout layers 842 can be implemented as a three-layer, batch-normalized (as described in further detail in Ioffee, et al., “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *Proceedings of the 32nd International Conference on Machine Learning*, 2015) maxout network (as described in further detail in Goodfellow, et al., “Maxout networks,” *Proceedings of the 30th Annual Conference on Machine Learning*, 2013) to produce a probability distribution over possible classes.

[0063] As discussed above and further emphasized here, Figure 8 is merely an example of a system for natural language processing which should not unduly limit the scope of the claims. One of ordinary skill in the art would recognize many variations, alternatives, and modifications. In some embodiments, system 800 may be modified so that it performs a different NLP task, such as, for example, question answering. For a NLP task of question answering, the task specific sequences x and y are obtained in the same way as for classification (Eq. 7 and Eq. 8), except that the function f is replaced with a function g that uses a tanh activation instead of a ReLU activation. In this case, one of the sequences is the document and the other is the question in the question-document pair. These sequences x and y are then fed through the coattention and dynamic decoder implemented, for example, as the Dynamic Coattention Network (DCN), as described in further in Xiong, et al., “Dynamic memory networks for visual and textual question answering,” *In Proceedings of the 33rd International Conference on Machine Learning*, pages 2397-2406, 2016.

[0064] Figures 9 and 10 are simplified diagrams comparing the performance of systems for natural language processing based on different input encodings. These Figures 9 and 10 illustrate how varying the input representations—e.g., GloVe alone, GloVe plus CoVe, GloVe plus Char, and GloVe plus CoVe plus Char—affects the final performance of NLP tasks such as sentiment analysis, question classification, entailment classification, and question answering.

[0065] Likewise, Figure 11 is a table illustrating performance results of systems for natural language processing based on different input representations (SST-2, SST-5, IMDb, TREC-6, TREC-50, SNLI, SQuaAD), and with different training sets (MT-Small, MT-Medium, and MT-Large) for the encoder (CoVe-S, CoVe-M, CoVe-L, respectively).

[0066] Figures 9 and 10 shows that models that used CoVe alongside GloVe achieved higher performance than models that used only GloVe. Figure 11 shows that using CoVe in Eq. 6 brings larger improvements than using character n-gram embeddings, as described in further detail in Hashimoto, et al., “A joint many-task model: Growing a neural network for multiple NLP tasks,” *arXiv preprint arXiv 1611.01587*, submitted on November 5, 2016. It also shows that altering Eq. 6 by additionally appending character n-gram embeddings can boost performance even further for some NLP tasks. This suggests that the information provided by CoVe is complementary to both the word-level information provided by GloVe as well as the character-level information provided by character n-gram embeddings.

[0067] Figures 9-11 validate the advantage or benefit of transferring knowledge from an encoder pretrained on machine translation to a variety of other downstream NLP tasks. In all cases, models that use context vectors (CoVe) performed better than baselines that used random word vector initialization, baselines that used pretrained word vectors from a GloVe model, and baselines that used word vectors from a GloVe model together with character n-gram embeddings.

[0068] Although illustrative embodiments have been shown and described, a wide range of modifications, changes and substitutions are contemplated in the foregoing disclosure and in some instances, some features of the embodiments may be employed without a corresponding use of other features. One of ordinary skill in the art would recognize many variations, alternatives, and modifications. Thus, the scope of the present application should be limited only by the following claims, and it is appropriate that the claims be construed broadly and in a manner consistent with the scope of the embodiments disclosed herein.

EMBODIMENTS IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A system for natural language processing, the system comprising:

a multi-layer neural network;

wherein the system is configured to:

convert at least one input sequence of words in a first language to a sequence of word vectors;

generate context-specific word vectors for the at least one input sequence of words in the first language using an encoder pre-trained using training data for translating phrases from the first language to phrases of a second language;

concatenate the word vectors and the context-specific word vectors; and

perform a first natural language processing task on the least one input sequence of words in the first language using the concatenated word vectors and context-specific word vectors.

2. The system of claim 1, wherein the first natural processing task is one of sentiment analysis, question classification, entailment classification, and question answering.

3. The system of claim 1, wherein the encoder comprises at least one bidirectional long-term short-term memory configured to process at least one of the word vectors in the sequence.

4. The system of claim 3, wherein the encoder comprises an attention mechanism configured to compute an attention weight based on an output of the at least one bidirectional long-term short-term memory.

5. The system of claim 1, wherein a decoder is used in the pre-training of the encoder, wherein the decoder is initialized with hidden vectors generated by the encoder.

6. The system of claim 5, wherein the decoder comprises at least one bidirectional long-term short-term memory configured to process at least one word vector in the second language during training of the encoder.

7. The system of claim 1, further comprising a biattentive classification network configured to generate attention weights based on the concatenated word vectors and context-specific word vectors.

8. A system for natural language processing, the system comprising:

an encoder for generating context-specific word vectors for at least one input sequence of words, wherein the encoder is pre-trained using training data for performing a first natural language processing task; and

a neural network for performing a second natural language processing task on the at least one input sequence of words using the context-specific word vectors, wherein the first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder.

9. The system of claim 8, wherein the first natural processing task is machine translation.

10. The system of claim 8, wherein the second natural processing task is one of sentiment analysis, question classification, entailment classification, and question answering.

11. The system of claim 8, wherein the encoder is pre-trained using a machine translation dataset.

12. The system of claim 8, wherein the neural network is trained using a dataset for one of sentiment analysis, question classification, entailment classification, and question answering.

13. The system of claim 8, wherein the first natural processing task is different from the second natural language processing task.

14. The system of claim 8, wherein the encoder comprises at least one bidirectional long-term short-term memory.

15. A method comprising:

using an encoder, generating context-specific word vectors for at least one input sequence of words, wherein the encoder is pre-trained using training data for performing a first natural language processing task; and using a neural network, performing a second natural language processing task on the at least one input sequence of words using the context-specific word vectors, wherein the first natural language process task is different from the second natural language processing task and the neural network is separately trained from the encoder.

16. The method of claim 15, wherein the first natural processing task is machine translation.

17. The method of claim 15, wherein the second natural processing task is one of sentiment analysis, question classification, entailment classification, and question answering.

18. The method of claim 15, wherein the encoder is pre-trained using a machine translation dataset.

19. The method of claim 15, wherein the neural network is trained using a dataset for one of sentiment analysis, question classification, entailment classification, and question answering.

20. The method of claim 15, wherein the first natural processing task is different from the second natural language processing task.

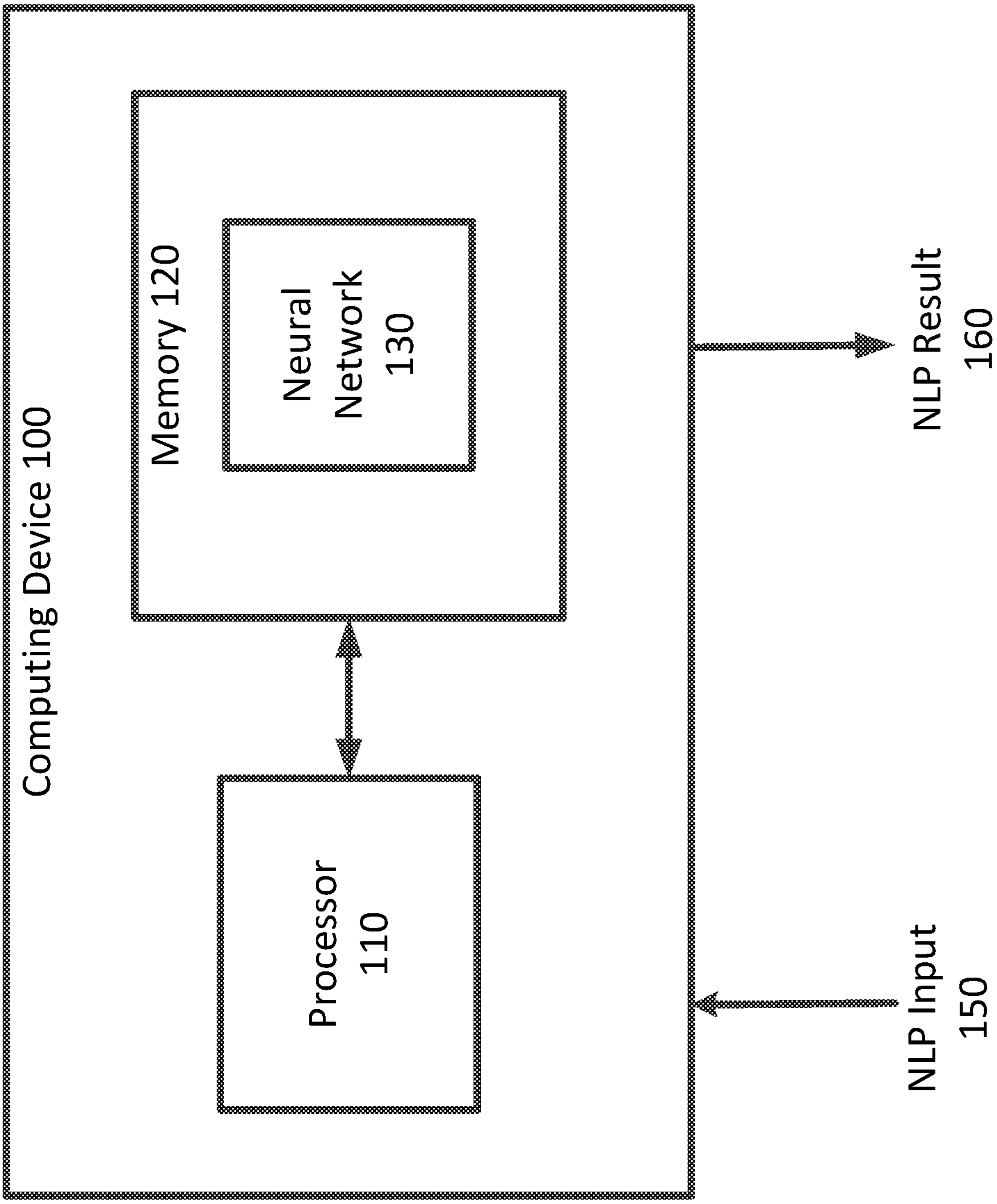

Computing Device 100
Memory 120
Neural Network 130
Processor 110
NLP Result 160
NLP Input 150

FIG. 1

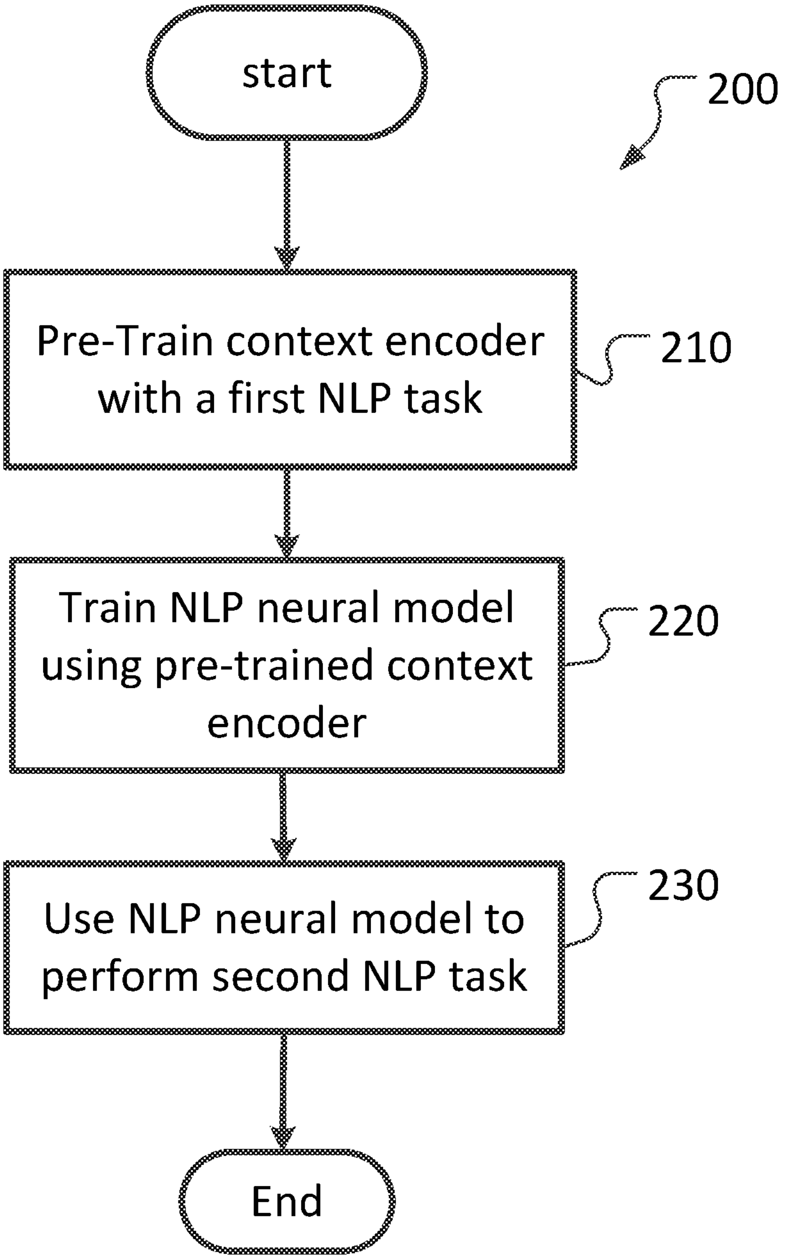

start
200
Pre-Train context encoder with a first NLP task
210
Train NLP neural model using pre-trained context encoder
220
Use NLP neural model to perform second NLP task
230
End

FIG. 2

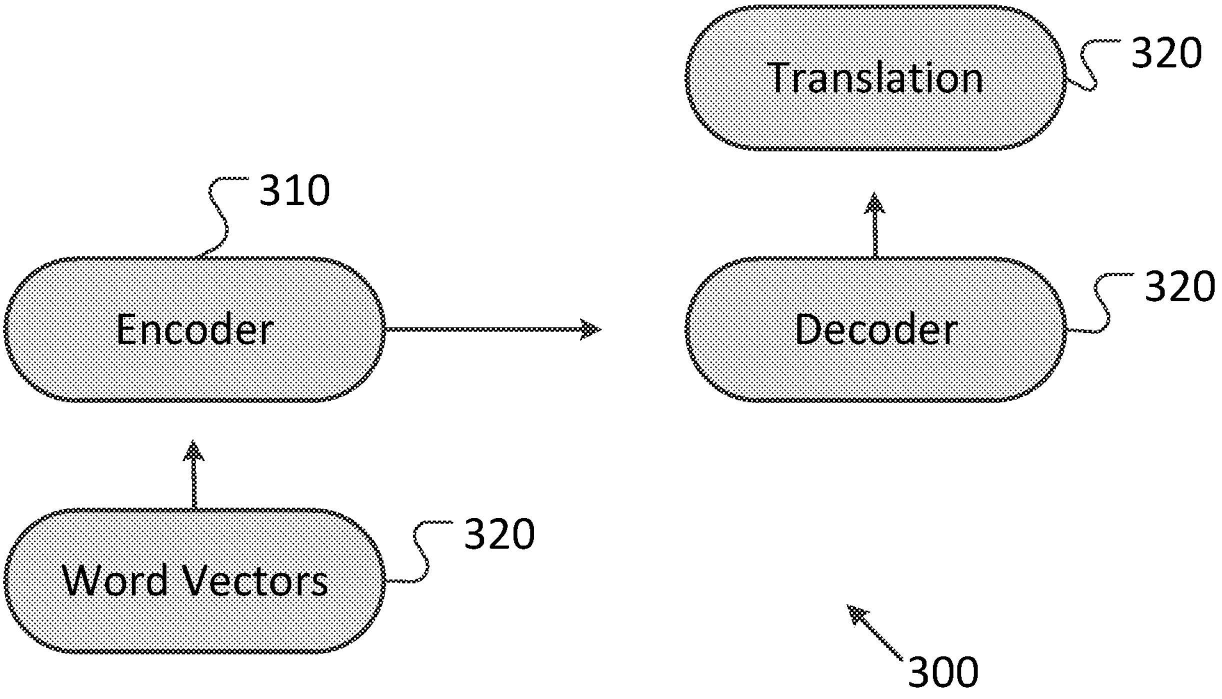

Translation
320
310
Encoder
Decoder
320
Word Vectors
320
300

FIG. 3

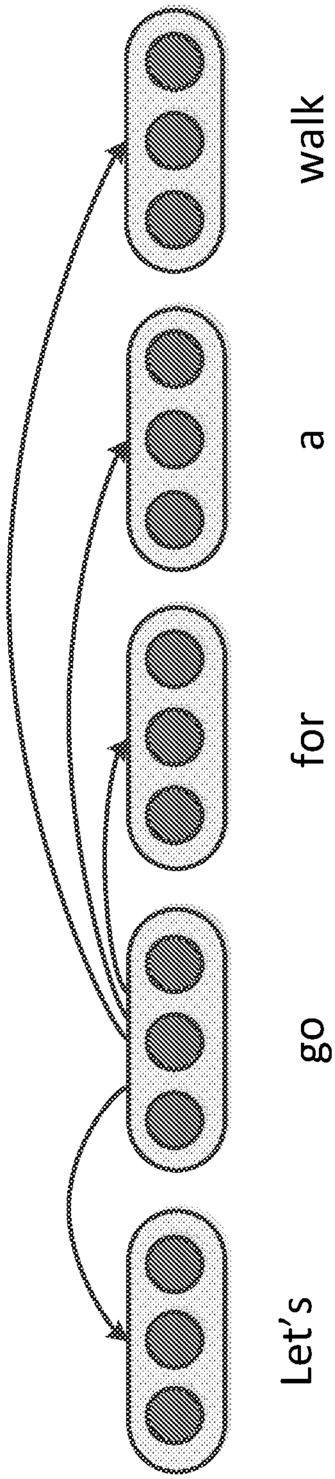


FIG. 4

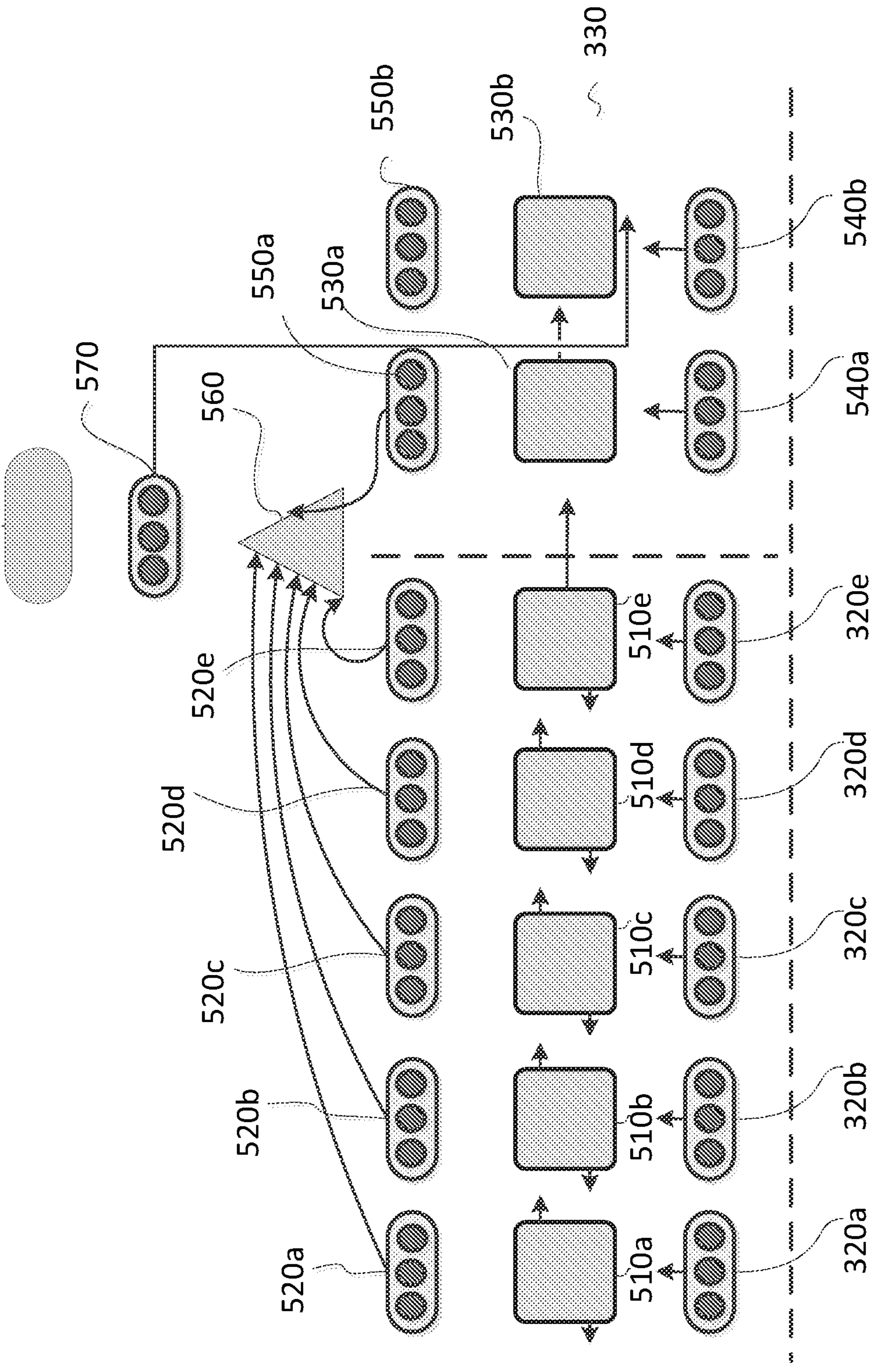

330
550b
530b
540b
550a
530a
540a
570
580
560
520e
510e
320e
520d
510d
320d
520c
510c
320c
520b
510b
320b
520a
510a
320a

FIG. 5

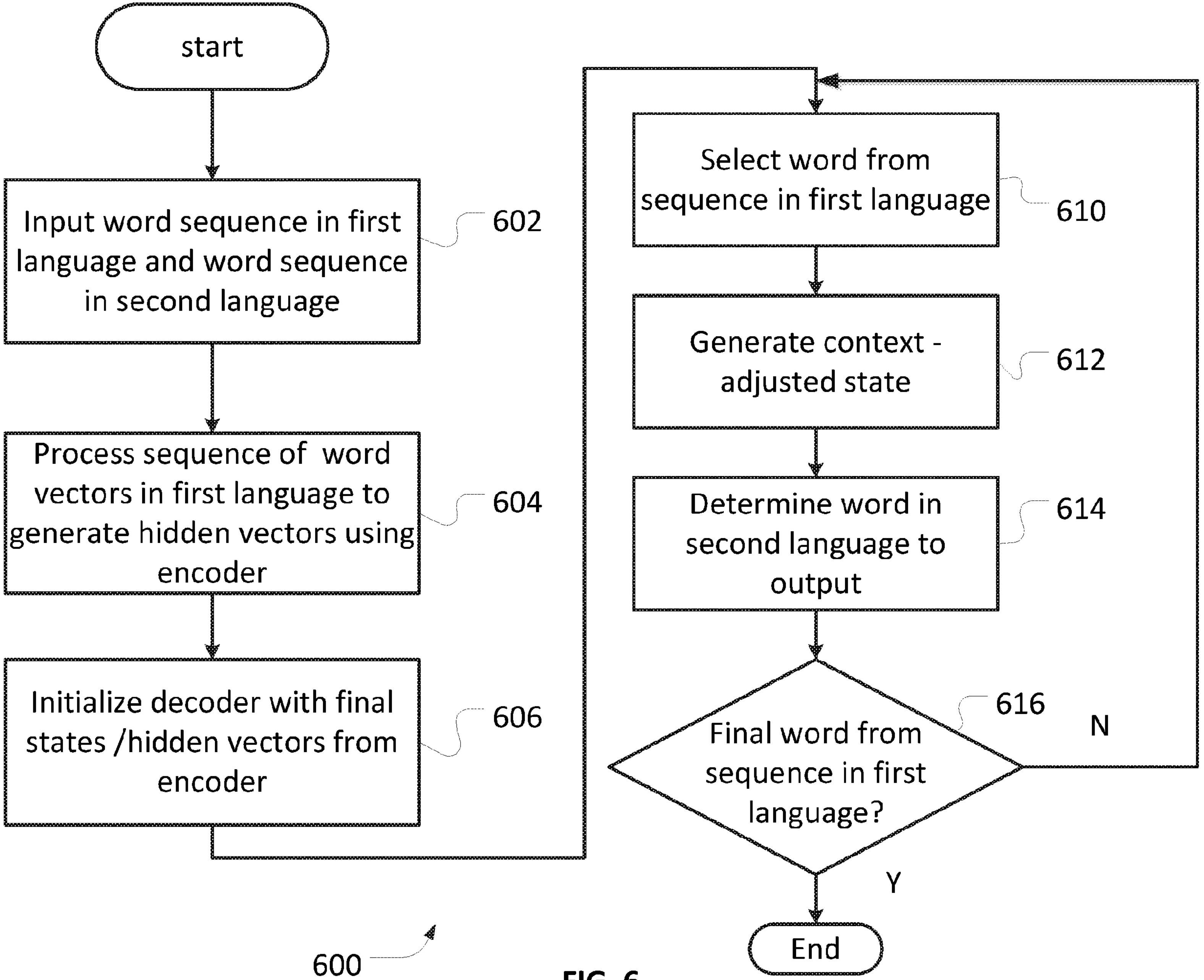

start
Input word sequence in first language and word sequence in second language
602
Process sequence of word vectors in first language to generate hidden vectors using encoder
604
Initialize decoder with final states /hidden vectors from encoder
606
Select word from sequence in first language
610
Generate context - adjusted state
612
Determine word in second language to output
614
Final word from sequence in first language?
616
N
Y
End
600

FIG. 6

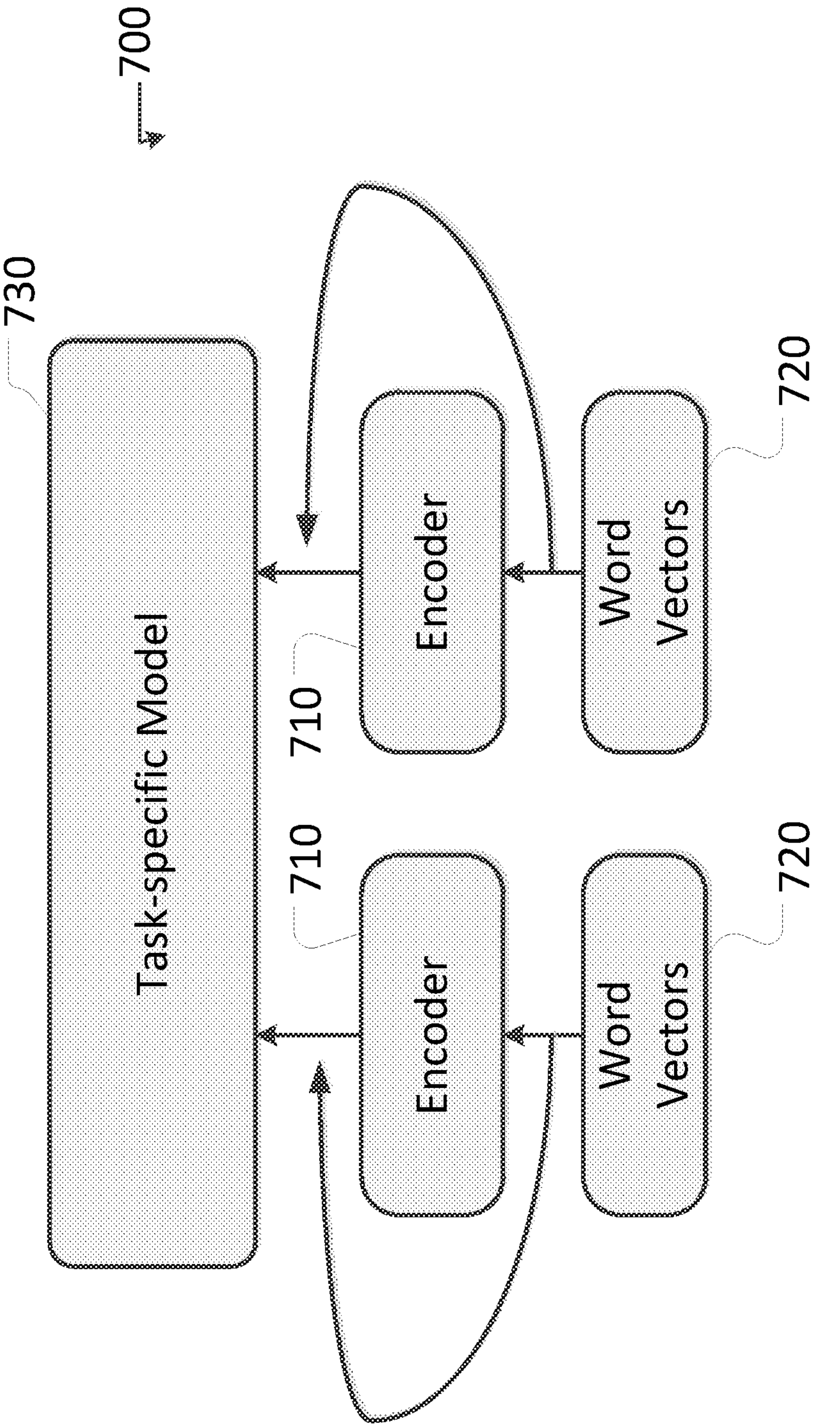

700
730
Task-specific Model
710
Encoder
710
Encoder
720
Word Vectors
720
Word Vectors

FIG. 7

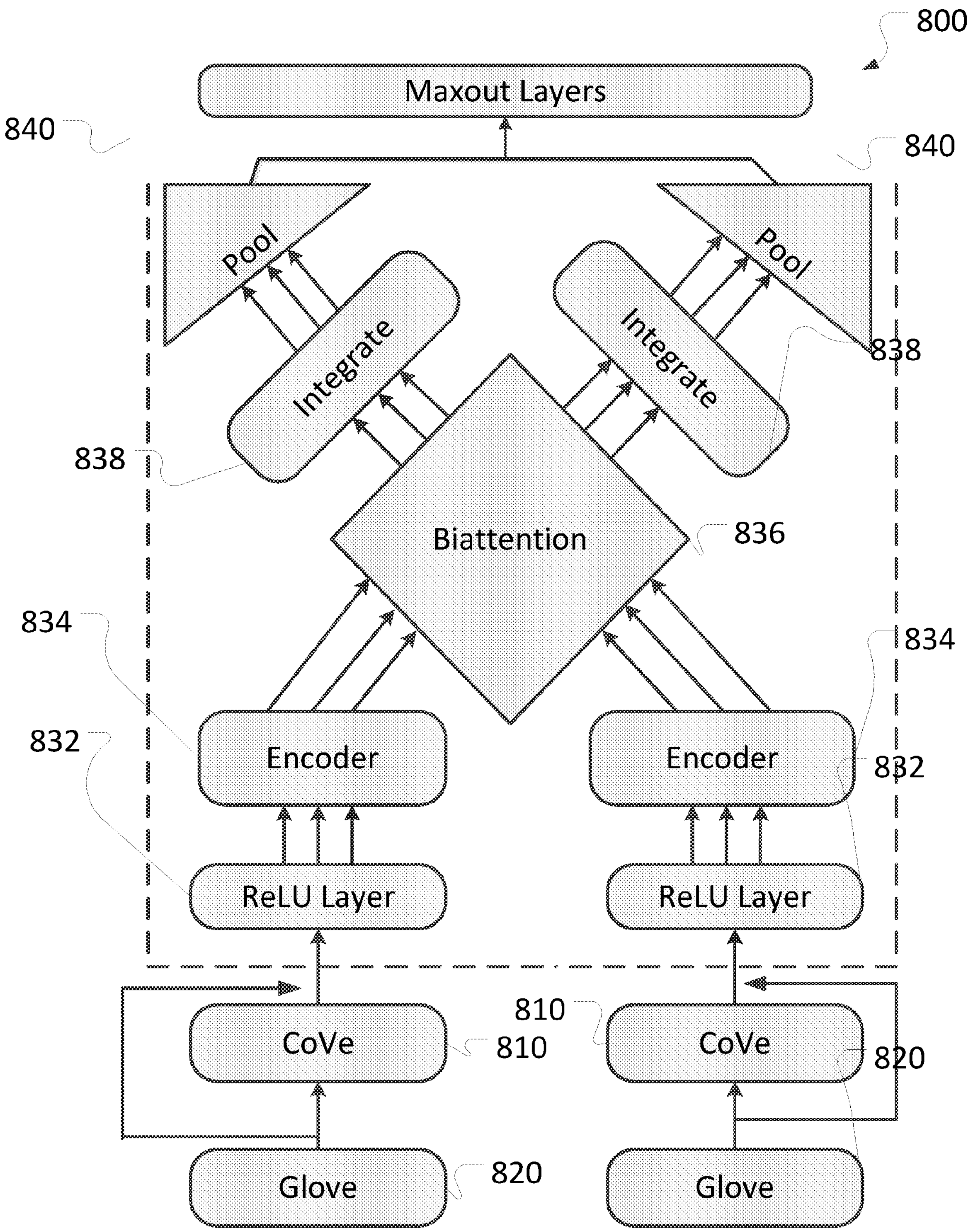

800
Maxout Layers
840
840
Pool
Pool
Integrate
Integrate
838
838
Biattention
836
834
834
Encoder
Encoder
832
832
ReLU Layer
ReLU Layer
810
CoVe
810
CoVe
820
820
Glove
Glove

FIG. 8

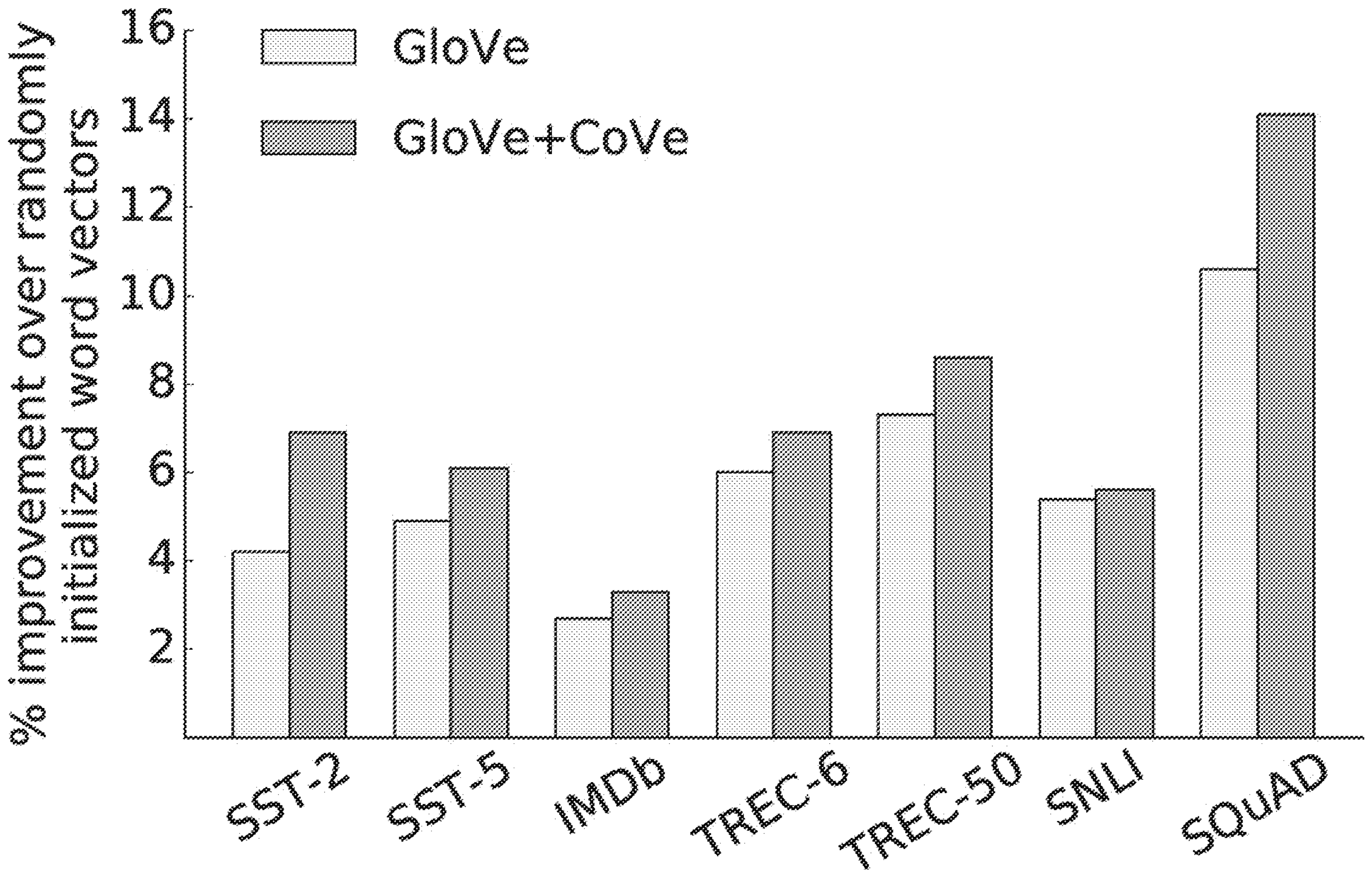

16
14
12
10
8
6
4
2
% improvement over randomly initialized word vectors
GloVe
GloVe+CoVe
SST-2
SST-5
IMDb
TREC-6
TREC-50
SNLI
SQuAD

FIG. 9

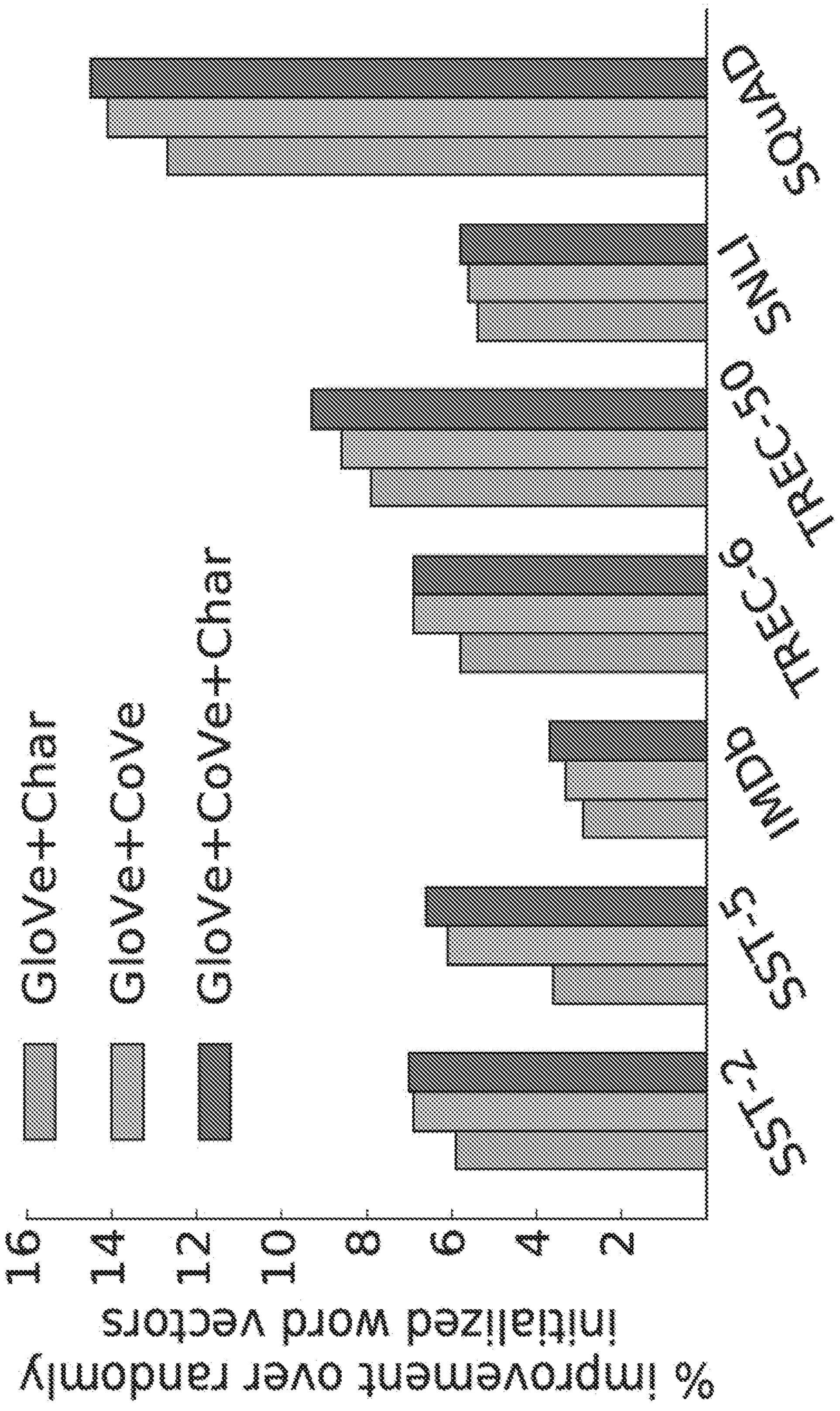
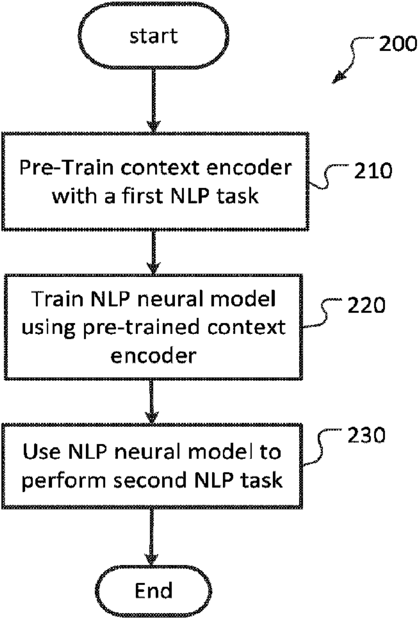

% improvement over randomly initialized word vectors
16
14
12
10
8
6
4
2
GloVe+Char
GloVe+CoVe
GloVe+CoVe+Char
SST-2
SST-5
IMDb
TREC-6
TREC-50
SNLI
SQuAD

FIG. 10

			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

FIG. 11


start
200
Pre-Train context encoder with a first NLP task
210
Train NLP neural model using pre-trained context encoder
220
Use NLP neural model to perform second NLP task
230
End