Microsoft ICECAPS: An Open-Source Toolkit for Conversation Modeling

Vighnesh Leonardo Shiv, Chris Quirk, Anshuman Suri, Xiang Gao,
Khuram Shahid, Nithya Govindarajan, Yizhe Zhang, Jianfeng Gao,
Michel Galley, Chris Brockett, Tulasi Menon, Bill Dolan
Microsoft Corporation
{vishiv, chrisq, ansuri, xiag, khurams,
nigovind, yizzhang, jfgao, mgalley,
chrisbkt, tulasim, billdol}@microsoft.com

Abstract

The Intelligent Conversation Engine: Code and Pre-trained Systems (Microsoft ICECAPS) is an upcoming open-source natural language processing repository. ICECAPS wraps TensorFlow functionality in a modular component-based architecture, presenting an intuitive and flexible paradigm for constructing sophisticated learning setups. Capabilities include multitask learning between models with shared parameters, upgraded language model decoding features, a range of built-in architectures, and a user-friendly data processing pipeline. The system is targeted toward conversational tasks, exploring diverse response generation, coherence, and knowledge grounding. ICECAPS also provides pre-trained conversational models that can be either used directly or loaded for fine-tuning or bootstrapping other models; these models power an online demo of our framework.

1 Introduction

Neural conversational systems have seen great improvements over the past several years, with current models able to generate surprisingly coherent dialogs (Gao et al., 2019a). Business applications, games, and potentially other settings can benefit from intelligent conversational agents, inviting users to interact intuitively with complex systems.

Although a range of open-source tools is available to train neural network models for natural language processing (Vaswani et al., 2018; Gardner et al., 2018; Klein et al., 2017), only a few emphasize multi-turn conversational settings (Miller et al., 2017; Burtsev et al., 2018). Conversations present distinct challenges. They generally consist of many turns, and agents need to contextualize responses in these multi-turn contexts. Agents may also need to contextualize their responses in other cues, such as style, intent, and external knowledge, while retaining a conversational flow.

We present ICECAPS[1], a conversation modeling toolkit developed to bring together these desirable characteristics. ICECAPS is built on top of TensorFlow functionality wrapped in a user-friendly paradigm. Users can build agents with induced personalities, capable of generating diverse responses, grounding those responses in external knowledge, and avoiding particular phrases. Our toolkit's foundation is an extensible framework based on composable model structures, supporting complex configurations with component chaining and multi-task training schedules. We also provide large pre-trained conversational systems to support fast exploration.

2 Architecture

ICECAPS is designed for modularity, flexibility, and ease of use. Modules are built on top of TensorFlow Estimators, making them easy for developers to use and extend flexibly. ICECAPS supports arbitrary architectures of modules chained together within versatile multi-task configurations.

2.1 Component chaining

Sequence-to-sequence models can be abstracted as chains of sequence encoders and sequence decoders. Our library implements various encoders and decoders, which can be chained together to form a single, end-to-end functional model. This chaining paradigm allows users to flexibly combine components and create topologies including multiple models with shared components. Chaining also allows users to bootstrap new models from components of previously trained models.

2.2 Multi-task learning

Multi-task learning is a powerful training paradigm that promotes robust feature represen-

[1] https://github.com/microsoft/icecaps

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 123–128
Florence, Italy, July 28 - August 2, 2019.

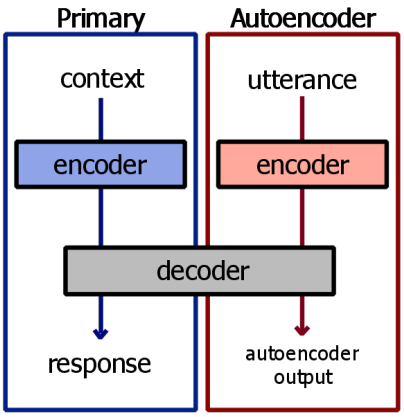

Figure 1: An example of a basic multi-task configuration. Two encoder-decoder chains share a common decoder, alternately trained on separate datasets.

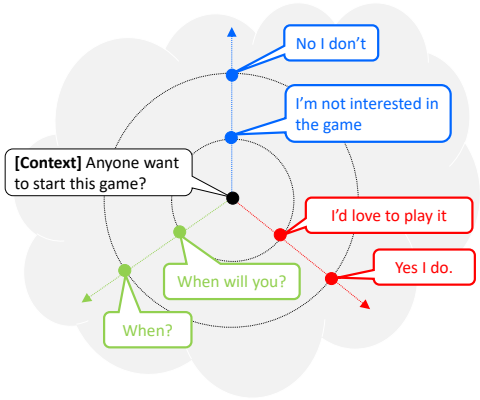

Figure 2: Illustration of latent space under SpaceFusion. The distance between predicted response vectors and contexts represents relevance; the angle between them represents intent. Figure from (Gao et al., 2019b).

tations (Gao et al., 2019b; Liu et al., 2019). By unifying a conversational sequence-to-sequence model and an autoencoder with a shared decoder, multi-task learning can personalize the conversational model (Luan et al., 2017). Multi-task learning has potentially many other powerful applications for inducing biases in conversational systems. ICECAPS allows users to build arrays of models with arbitrary sharing of components, and place them in a multi-task learning environment. Users can construct arbitrary multi-task training schedules, assigning different tasks or balances among tasks per training step.

3 Built-in modules and configurations

ICECAPS provides several built-in modules and configurations. Most standard NLP architectures are available, including transformers (Vaswani et al., 2017), LSTM-based seq2seq models (Sutskever et al., 2014) with attention (Bahdanau et al., 2015; Luong et al., 2015), n-gram convolutional language models, and deep convolutional networks for baseline image grounding. Where applicable, these are implemented as chains of simpler components as per our design philosophy. We also provide features that target conversational scenarios, from individual chainable components to custom multi-task learning presets.

3.1 Personality grounding

Inspired by recent work on modeling personality differences in conversational systems (Li et al., 2016b), ICECAPS provides implementations of personality-grounded seq2seq and transformer decoders. This architecture consists of additional personality embeddings, which are provided to the decoder alongside token embeddings at each timestep of decoding. Grounding generated responses helps condition outputs based on a given personality embedding: for the same query, the system learns to generate responses in different styles, all while preserving the underlying context.

3.2 SpaceFusion

SpaceFusion (Gao et al., 2019b) is a learning paradigm that aligns latent spaces learned by different models trained over different datasets. Of particular interest is its application to neural conversation modelling, where SpaceFusion can optimize the relevance and diversity of generated responses jointly. ICECAPS implements a SpaceFusion preset that extends its multi-task capabilities. SpaceFusion constructs a multi-task environment of two seq2seq models with a shared decoder, as in (Luan et al., 2017). It distinguishes itself by modifying the multi-task objective function with several regularization terms. These extra terms encourage responses for the same context to be placed nearby in latent space and aligning semantically related responses along straight lines in latent space. This induces a structure in the latent space such that distance and direction from a predicted response vector roughly correspond to relevance and diversity, respectively, as in Figure 2.

3.3 Knowledge grounding

A critical task in building intelligent conversational agents is grounding their responses in an external knowledge base. This allows agents to

provide informed responses with context about the real world, without needing comprehensive paired conversational data to embody that information. We provide an extension of stochastic answer networks (Liu et al., 2018), a machine reading comprehension system, that acts as a full knowledge-grounded conversation model (Qin et al., 2019), hybridizing machine reading comprehension with a response generation model. At a high level, this model consists of two deep biLSTMs in parallel that encode conversational context and knowledge, respectively. The information from these encoders is then combined using cross-attention, the output of which forms the basis of a memory cell that powers a response generator.

4 Decoding features

ICECAPS provides a custom beam search decoder that extends TensorFlow's native beam search decoder by introducing several useful features.

4.1 Diverse generation with MMI re-ranking

Generative language systems are notorious for generating bland, uninteresting samples. Although generated hypotheses generally have high scores on metrics used to approximate context and relevance of generated texts (e.g. perplexity and BLEU), these metrics fail to measure diversity, a highly desirable trait for responses generated by conversational systems.

To alleviate this, we provide an implementation of the maximum mutual information (MMI) scoring function (Li et al., 2016a). We extend and modify the TensorFlow TrainingHelper and BasicDecoder classes for our implementation. MMI employs a separately trained model that learns to predict queries from given responses: the inverse map of the conversational model. Using the MMI model, for a given set of hypotheses, we calculate the log-probability of a given query per hypothesis: $P(query|hypothesis)$. This approximates response diversity, as frequent and repetitive hypotheses would be associated with many possible queries, thus generating a lower probability for any specific query. This score is weighted and used to re-rank hypotheses, pushing blander responses below context-unique responses.

4.2 Token filtering

Models trained on real-world data may utter undesirable words or phrases. Users may want the agent to avoid profanities or other offensive language. Likewise, the system should avoid obvious ungrammatical outputs, such as broken abbreviations or nonsensical punctuation marks.

ICECAPS supports several filters, including a general censor-list and a start-token censor-list. The general censor-list contains a list of tokens to disable during response generation; probabilities associated with these tokens are clamped to zero. The start-token censor-list is similar, but only masks the response's first token. We also support infrequency filters; users may restrict the decoder from generating responses with rare words.

4.3 Modified beam search decoding

The standard beam-search implementation in TensorFlow works by iteratively generating tokens, generating a constant number of hypotheses at the end of the decoding phase. ICECAPS implements a modified beam search decoder with a different criterion for exploring complete hypotheses. Rather than considering a completion of every hypothesis, this decoder only considers a complete version of a hypothesis if the END token is one of the top k options for the next token. This version of beam-search decoding may result more more or less than k final hypotheses, depending on how often the decoder produced an END token.[2] This form of decoding can sometimes produce cleaner hypotheses than the standard beam-search implementation, perhaps because END is only allowed when the model score is high. This helps increase the quality of generated sequences, as they tend to have improved grammatical coherence, though the number of returned outputs is often less than k.

4.4 Repetition penalty

The ICECAPS custom decoder also includes a repetition penalty, used in the scoring function employed during the beam search phase. This penalty helps avoid the well-known problem of decoders generating repetitive responses and getting stuck in loops. The repetition penalty is calculated as:

$$\log\left(\min\left(1, \frac{uniq(s)}{\lceil(A \times \|s\|)\rceil}\right)\right) \quad (1)$$

for a given response s, where A is the repetition allowance and $uniq(s)$ is the number of unique tokens in s.

[2] If no responses are generated by the last time-step, we return all hypotheses generated in the last time-step, ensuring that the decoder always produces at least one response.

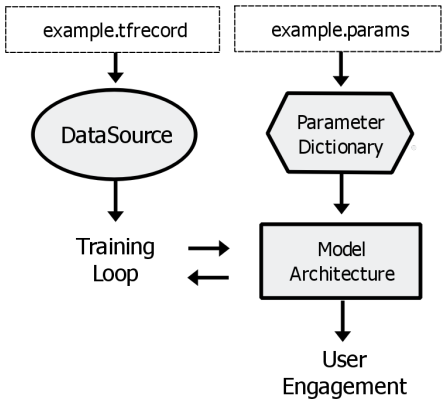

Figure 3: The five-phase pattern underlying ICECAPS training configurations. Hyperparameters are extracted into a dictionary from a file, which are used to initialize the model architecture. Data files initialize DataSource objects, which feed the training loop. The user can engage with the system once trained.

5 Building systems

ICECAPS is designed to make building complex dialogue systems intuitive for the end user.

5.1 Text data processing

TensorFlow estimators expect to read data from TFRecord binary files for efficient processing. We provide a script TEXT_DATA_PROCESSING.PY for converting text data into TFRecords, equipped with several useful preprocessing transformations. Our script can sort data within local windows so that batches fed during training have minimal padding inefficiency. These batches can be shuffled amongst each other to mitigate any biases induced by sorting. We provide token preprocessing through byte pair encoding (Sennrich et al., 2016), which builds a token set at a level of abstraction between characters and words. This often allows for faster training and improved generalization. Another feature focused on conversational scenarios is fixed-length context extraction. Conversational data often contains large, potentially unwieldy multi-turn contexts; we can limit our data samples to a desired context length. We also provide an option for annotating datasets with topic grounding information, by analyzing the data for unique tokens to use as topic markers.

5.2 Training configurations

ICECAPS training configurations follow a basic five-phase pattern. We include example training scripts that users may use as templates.

Loading hyperparameters. We supply hyperparameters to the model architecture via .PARAMS files. These files follow a simple, readable format: each line pairs the name of a hyperparameter with its supplied value, separated by a colon. Every ICECAPS estimator is equipped with a set of expected hyperparameters, and their associated default values when those hyperparameters are not provided. Users can view these hyperparameters through a static API call. In systems with multiple modules, users can add prefixes to hyperparameters to properly assign them to the right modules.

Building model architecture. Architectures are built by composing modules, chaining them and placing them in EstimatorGroups as appropriate. They are initialized with the loaded dictionaries of hyperparameters.

Connecting data sources. We now connect TFRecord files to our architecture via an input pipeline. We provide a high-level wrapper around TensorFlow's native Dataset-based pipelining. Users initialize DataSource objects with TFRecord files and produce input pipelines from those files. Our DataSource class also provides methods for combining multiple input pipelines from DataSource objects, either by randomly interleaving them or by combining them in parallel to feed multi-task configurations.

Training the system. The system is now ready to train: a single call to the training function with the desired number of batches or epochs is sufficient. However, users can construct more elaborate training schedules, consisting of sequences of training function calls with different arguments. This is particularly useful in multi-task scenarios, where we may want to train the system with different balances across tasks in different steps. For instance, one could first pre-train a single task, then shift to multi-task learning evenly distributed across two tasks.

The user can now engage with the trained system. Users can run their system on an evaluation set, collecting appropriate metrics and decoded responses. They can also interact with the system directly. ICECAPS provides a command-line interactive session for users to have conversations with their agents and directly observe their responses. Response generation is powered by the custom decoder described in Section 4. While the command-line session is useful for quick testing, for conve-

nience we also provide a simple GUI-based interactive session in which users can load their trained models. The GUI makes it easy to view multiple turns of conversation history alongside a top-k list of generated responses with associated scores.

6 Pre-trained models

ICECAPS comes packaged with pre-trained systems for conversation modeling based on the features described in Section 3. Users can either employ these systems to load conversational agents for immediate use or to bootstrap the training process for new configurations.

The largest-scale pre-trained system we currently offer is a deep Transformer-based architecture trained on real-world conversational data. Our model employs 12 layers with layer normalization, a modified initialization scheme that accounts for model depth, and byte pair encodings (Sennrich et al., 2016) for the tokenizer. We trained this system on a large corpus of conversations scraped from Reddit. The data was extracted from Reddit comment chains spanning from 2005 till 2017. The dataset consists of hundreds of millions of paired instances of contexts and responses with billions of tokens. Our model uses a vocabulary size of 50,257, and was trained on eight Nvidia V100 machines with NVLink.

We provide a set of trained personality embeddings for implementing diverse personality chatbots. These embeddings were learned through multi-task learning between paired conversation data and unpaired utterances categorized by speaker. These embeddings define a personality space; users may use the provided embeddings for their applications or train new personality embeddings within this space.

We also provide a demo of a conversational agent that combines a number of key features discussed in this paper. This agent is powered by an LSTM-based seq2seq model built on the SpaceFusion paradigm. Our agent demonstrates the improvements to conversational response generation made possible by a combination of multi-task learning and our improved beam search decoder.

7 Related toolkits

Several NLP-oriented toolkits have been open-sourced. Tensor2Tensor (Vaswani et al., 2018), maintained by Google Brain, extends TensorFlow with an array of state-of-the-art baseline deep learning models. It places a strong emphasis on sequence modeling baselines. AllenNLP (Gardner et al., 2018) is a PyTorch library developed by AI2 for natural language processing tasks, notable for an open-source release of ELMo (Peters et al., 2018). OpenNMT (Klein et al., 2017) is a popular neural machine translation toolkit originally developed for LuaTorch that now has implementations in PyTorch and TensorFlow. MarianNMT (Junczys-Dowmunt et al., 2018) is another framework for neural machine translation developed between the Adam Mickiewicz University in Pozna and the University of Edinburgh. It is built in C++ and designed for fast training in multi-GPU systems. Texar (Hu et al., 2018) is a text generation toolkit affiliated with Carnegie Mellon University, featuring a similar emphasis on modularity to ICECAPS. It includes reinforcement learning capabilities alongside its sequence modelling tools.

A few other toolkits have a dialog emphasis. DeepPavlov (Burtsev et al., 2018) is a deep learning library with a focus on task-oriented dialogue. It provides demos and pre-trained models for tasks such as question answering and sentiment classification. Affiliated with DeepPavlov is the ConvAI2 challenge (Dinan et al., 2019), a general dialogue competition featuring a synthetic personalized conversational dataset. ParlAI (Miller et al., 2017) is a library centered around task-oriented dialogue, compiling a number of popular datasets for NLP tasks as well as pre-trained models for knowledge-grounded dialog agents trained on crowd-sourced data.

8 Conclusion

Microsoft ICECAPS is a new open-source NLP library focused on building intelligent conversation agents that can communicate naturally with humans. Our release contributes key conversation modeling features to the open-source community, including personalization, knowledge grounding, diverse response modeling and generation, and more generally a multi-task architecture for inducing biases in conversational agents. Built for modularity and ease of use, ICECAPS allows users to extend our conversational technologies in novel ways for their agents. We provide the community with a number of pre-trained conversational systems trained on real-world data. Planned future directions for ICECAPS include multi-modality grounding and semantic parsing.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

M. Burtsev, A. Seliverstov, R. Airapetyan, M. Arkhipov, D. Baymurzina, N. Bushkov, O. Gureenkova, T. Khakhulin, Y. Kuratov, D. Kuznetsov, A. Litinsky, V. Logacheva, A. Lymar, V. Malykh, M. Petrov, V. Polulyakh, L. Pugachev, A. Sorokin, M. Vikhreva, and M. Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics-System Demonstrations*.

E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, J. Williams, J. Pineau, M. Burtsev, and J. Weston. 2019. The second conversational intelligence challenge (convai2).

J. Gao, M. Galley, and L. Li. 2019a. Neural approaches to conversational AI. *Foundations and Trends® in Information Retrieval*.

X. Gao, S. Lee, Y. Zhang, C. Brockett, M. Galley, J. Gao, and B. Dolan. 2019b. Jointly optimizing diversity and relevance in neural response generation. In *NAACL-HLT 2019*.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software*.

Z. Hu, H. Shi, Z. Yang, B. Tan, T. Zhao, J. He, W. Wang, L. Qin, D. Wang, et al. 2018. Texar: A modularized, versatile, and extensible toolkit for text generation. *arXiv preprint arXiv:1809.00794*.

M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins, and A. Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018 System Demonstrations*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of NAACL-HLT*.

J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of ACL*.

X. Liu, P. He, W. Chen, and J. Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

X. Liu, Y. Shen, K. Duh, and J. Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Y. Luan, C. Brockett, B. Dolan, J. Gao, and M. Galley. 2017. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*.

T. Luong, H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. ParlAI: A dialog research software platform. In *Proceedings of the 2017 EMNLP System Demonstration*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

L. Qin, M. Galley, C. Brockett, X. Liu, X. Gao, B. Dolan, Y. Choi, and J. Gao. 2019. Contentful neural conversation with on-demand machine reading. In *ACL 2019*.

R. Sennrich, B. Haddow, and A. Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of ACL (Volume 1: Long Papers)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc.

A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in NIPS 30*. Curran Associates, Inc.