Eigen: A Step Towards Conversational AI

William H. Guss
Machine Learning at Berkeley
University of California, Berkeley
wguss@ml.berkeley.edu

James Bartlett
Machine Learning at Berkeley
University of California, Berkeley
james@ml.berkeley.edu

Phillip Kuznetsov
Machine Learning at Berkeley
University of California, Berkeley
philkuz@ml.berkeley.edu

Piyush Patil
Machine Learning at Berkeley
University of California, Berkeley
ppatil@berkeley.edu

Abstract

In this document we outline the data collection methods, system, and experiments we ran during the course of this project. We aim to shed light on what drove the development of our bot and the techniques we used to power our bot in the course of the conversation.

1 Introduction

Automated dialogue systems have been a holy grail in the artificial intelligence research community. Dialogue systems already form the basis behind many customer support hotlines, chatbot user interfaces, and digital personal assistants like Amazon Alexa. Despite the sophistication of these technologies, their backbone relies exclusively on a set of predefined rules, preventing these technologies from scaling without significant human intervention. The reemergence of neural networks at the beginning of the decade holds promise towards improving the state of these dialogue agents, as neural networks have accelerated technologies in the domains of image recognition [Krizhevsky et al., 2012], machine translation [Wu et al., 2016], and speech recognition [Hannun et al., 2014]. These models also hold promise for generative dialogue models, as shown in Serban et al. [2016b] and Serban et al. [2016a].

Despite the success of these models, they require extremely large and varied datasets to mimic typical human speech and often succumb to mode-collapse, despite attempts to add stochasticity to these models. To alleviate this issue with generative dialogue models, we propose a method of categorizing turns of conversation into what we call Modes of Conversation (MOC). We define an MOC as the building block to turn-by-turn conversation. Our system attempts to avoid the problems of mode collapse by creating generative models for every mode of conversation, while also using a discriminator on the input sentence to find the appropriate generative model for a specific task. By reducing the scope of a generative model to a single mode of conversation, we significantly reduce the complexity that a single generative model needs to handle, allowing for easier incremental improvements in the system than with an entirely end-to-end learning system.

1.1 Overview and Goals

The core idea behind the Eigen system is that conversation can be broken into several different modes of conversation (MOCs). A mode of conversation dictates the kinds of information our bot needs to know to formulate a response, as well as the types of future responses we predict might occur in the next few turns of the conversation. Furthermore, in a conversation, we can bin each turn or set of

1st Proceedings of Alexa Prize (Alexa Prize 2017).

turns into one of these modes and predict the most appropriate next mode given a list of previous utterances in a conversation. In combination, the Eigen system combines expert level modules in each mode by virtue of the discriminator in addition to a joint optimization between the modules.

1.2 Modes of Conversation

Modes of conversation are the most essential unit of modular computation in the Eigen system. The following high level MOCs serve essentially to provide a very fine description of different modal interactions in conversation and not directly the modules which best act to encapsulate each mode. The MOCs we used are: *answer, clarification, conversational_formality, declaration, empathy, humor, identity, influence, other, question, and understanding.*

2 Data

Our first goal was to acquire a broad range of conversational data. As such, we created a monolithic corpus by collating and standardizing numerous corpora from the natural language generation literature. A list and short description of each dataset is provided below.

Conversational Datasets Collated

- **Switchboard** Switchboard (Godfrey and Holliman [1993]) is a collection of two-sided phone calls. We utilized the transcripts of these phone calls. This dataset has very relevant data, and unlike other datasets has only two-sided conversations. However, the dataset is relatively small.
- **r/ChangeMyView** r/ChangeMyView is a subreddit community focused on having civilized arguments on controversial topics. We collected comment data from this subreddit. Whilst the data is mostly high quality text, it does suffer from profanity issues at times and is a comment thread and not a two-sided conversation, which limits is utility.
- **Reddit Comment Corpus** This corpus is similar to the r/ChangeMyView one, except it is a collection of all comments on all subreddits on Reddit, rather than just r/ChangeMyView. While this provides a huge collection of conversational data, it suffers severely from malformed and profane text, and much of the conversations are not on relevant topics. Additionally, it is not structured in a two-sided conversational format.
- **Supreme Court Dialogue Corpus** The Supreme Court Dialogue Corpus (Danescu-Niculescu-Mizil et al. [2012]) is a collection of transcripts from Supreme Court cases. While this text is well-formed and eloquent, it is not conversational as it is mostly formal arguments.
- **Internet Argument Corpus** The Internet Argument Corpus (Walker et al. [2012]) is a collection of arguments from various internet forums. Much like r/ChangeMyView data, it is well-formed and generally fairly eloquent, however it is not a two-sided conversation.
- **Cornell Movie Dialogs Corpus** The Cornell Movie Dialogs Corpus (Danescu-Niculescu-Mizil and Lee [2011]) is a collection of movie transcripts from various blockbuster films. This data is fairly well-formed and for the most part very conversational, however it is not a two-sided conversation.
- **Santa Barbara Corpus of Spoken American English** The Santa Barbara Corpus of Spoken American English (Du Bois et al. [2000-2005]) is a corpus of various spoken interactions from across the US. It is very diverse and is mostly structured two-sided conversations, however many of the topics are not very relevant to colloquial conversation and it is a relatively small corpus.
- **Ubuntu Dialogue Corpus and Ubuntu Chat Corpus** Ubuntu Dialogue Corpus and Ubuntu Chat Corpus (Lowe et al. [2015]) is a corpus of conversations collected from Ubuntu IRC support forums. Although it is very colloquial and is a very large corpus, it is almost exclusively focused on technical support which is not very relevant and suffers from the presence of rare pronouns and URLs.

In order to facilitate transfer learning, we saw it necessary to also collate English language datasets. This way we can pretrain models on a huge English language dataset, in the hopes that the models

would learn English from this dataset and then could use the conversational datasets just to alter their understanding of English to fit in with conversational norms. A list of each English language dataset with short descriptions is provided below.

English Language Datasets Collated

- **British National Corpus** The British National Corpus (Clear [1993]) is a collection of 100 million words from various sources of English language.
- **Google Books Corpus** The Google Books Corpus is a collection of English language sentences from various books.
- **Wikipedia Data Dump** This corpus contains every single English Wikipedia page. We removed all non-text pieces of the pages.
- **Blog Authorship Corpus** The Blog Authorship Corpus (Schler et al. [2006]) consists of almost 20000 different blog authors' blog posts from a single year. This provides a large dataset of fairly colloquial English language.
- **Project Gutenberg** This corpus contains every single English book available on Project Gutenberg (a project to provide free books).

2.0.1 Preprocessing

To train models on a variety of conversation and textual data, it's necessary for the data to conform to one shared schema, depicted in the next section. To process the above datasets to fit this schema, we make use of Python's Natural Language Toolkit (NLTK). We tokenize raw text into paragraphs, and further decompose each paragraph into sentences which are then tokenized into words and punctuation. This tokenization process is fast and simple - paragraphs are delimited by newline characters, and sentences by standard sentence disambiguators such as periods, not part of abbreviations, followed by capital letters. We also make use of NLTK's named entity recognition capabilities to replace named entities with a special identifying token, to enforce the model's avoidance of learning lexemes or semantics based on specific, individual entities rather than viewing them as syntactical placeholders.

2.0.2 Standard Format

In order to facilitate easier processing during training of models, we transformed all datasets into a standard format. We used different schemas for conversational data and English language data. For conversational data, we stored each turn of conversation in a list where one turn is one speaker talking. The conversational schema is shown in Listing 1 in the Supplementary Materials. For English language data, we stored a list of paragraphs where each paragraph is represented by a list of sentences in that paragraph. The English language schema is shown in Listing 2 in the Supplementary Materials. Both schemas store sentences as both raw text and as tokens. These tokens are described above in Section 2.0.1. Additionally, for conversation data the MOC labels, acquired through Mechanical Turk as described in Section 2.2, are appended to each turn.

Although we collated numerous datasets into a larger corpus, we realized that many of these corpora did not represent the broad spectrum of standard two-party conversation because they were either focused in a niche area (Ubuntu corpus) or were not a structured two party conversation (Reddit data). We believed that the best path forward would be to develop such a dataset of natural two party conversations. We desired to develop a crowd-sourcing solution to this problem by exploring two directions - Bot Or Not and Mechanical Turk.

2.1 Bot or Not

Bot or Not is a website we developed to collect dialogue data. We built out a lightweight chat service that would connect one user to another "dialogue agent". This dialogue agent could either be a chat bot or another human. The user's objective is to hold a conversation with the agent and determine based on that conversation whether their chat partner was a chatbot or a human, and were prompted to vote "Bot" or "Not" at the end of their conversation.

Motivation We wanted to create an experience that would be enjoyable for users. We could have easily framed this as a Mechanical Turk experiment, following the data collection method of the

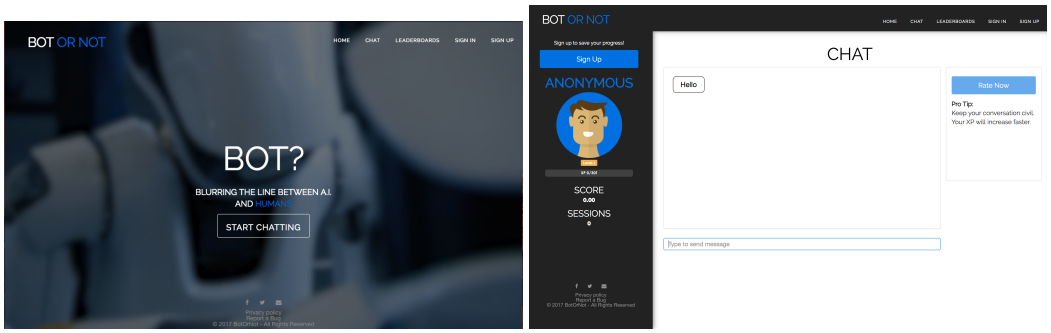

Figure 1: Left: The Bot or Not landing page. Right: A sample chat page from Bot or Not. Live demo at `http://beta.botornot.ml/`

Visdial Dataset [Das et al., 2016]. However, we believed this method is rather slow, expensive, and yields a limited amount of data. Our goal of making an inverse reinforcement learning system would require a large variety of dialogues, and Mechanical Turk would not be feasible in this specific context.

Instead we looked towards the crowd-sourcing techniques used by Galaxy Zoo [Willett et al., 2016] and QuickDraw [Ha and Eck, 2017]. These techniques successfully sourced extremely large datasets, facilitated by the ease of the task, the gamified experience, and their viral releases to the public.

Experience We aimed to mimic the simplicity and the wide reaching audience of [Willett et al., 2016] and [Ha and Eck, 2017]. We made entering a chat room the most visible part of the landing page, meaning a user could load the page and start chatting within seconds. A chat room could either start empty or with a prompt related to one of the five conversation domains. The users would then chat for 5 turns, at which point the "rate now" button in the top right of the screen would become clickable and a user could end the conversation and input their guess for whether their conversation partner was a bot or not. Upon selecting an option, the user would receive a notification of what the other user rated them as and whether or not they correctly guessed their partner's identity.

Gamification We looked towards gamification techniques that utilized a level system along with leaderboards to encourage users to yield more conversations, return to the website, and encourage them to share the website with their friends. Mekler et al. [2013] finds that websites using gamification elements like points, levels, and leaderboards positively increase the engagement of users in an image annotation task. To mimic these results, we created a points system based on the number of conversations a user had and the length of those conversations. The user was also further rewarded for guessing the identity of their conversation partner correctly. As a caveat, however, we randomized the label of the conversational partner. We wanted to ensure that our users would believe that we had created very convincing artificial intelligence - otherwise they would be inclined to believe that every conversational partner was a human and would quickly finish playing the game. However, we realized that users would only continue playing if they had a notion that they were getting better at the game. We thus set the probability of changing a conversational users' identity such that it was inversely related to the user's level. This way, a user with a higher level would believe that they are actually improving at the game.

Evaluation Framework Aside from data collection, Bot or Not can also function as a chat bot evaluation framework. We ran an early version of our conversational bot on the platform as a way to get a metric for the quality of our bot. For every bot we ran, we collected bot/not rating percentages and could use this information to compare the conversational appeal between two bots - a bot that could fool more users into believing that it was human would likely also perform better in the conversational settings.

Reward Signal In addition to using Bot or Not for evaluation, we have considered using the bot/not ratings on our chatbot as our reward signal for reinforcement learning. We have yet to use these results for training.

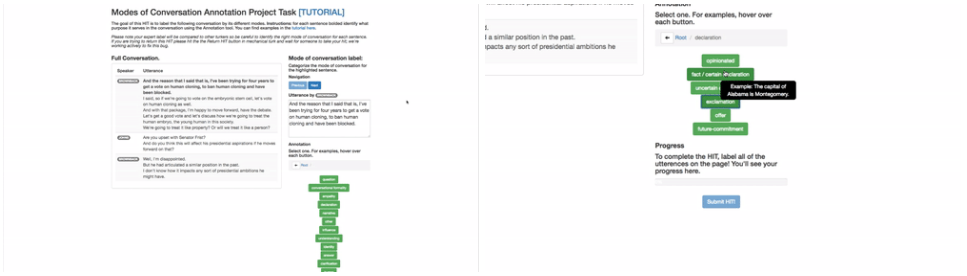

Figure 2: Left: The main screen seen by Turkers in the EigenTurk task. Right: A menu for choosing the sub-MOC for the bolded utterance on the main screen.

2.2 MOC Annotation

In order to achieve the desired partitioning of conversational action space into modes of conversation, the corpora of conversation data collected through Bot or Not and our data aggregation needed to be labeled according to their particular mode of conversation. Given the scope and the scale of the data, we considered several techniques to label the modes of conversation in each turn.

Structured Labeling The first mechanism relies on an ensemble of weak learners. In this regime, several hundred sparse, heuristic labeling functions are applied to the data to generate a partitioning of the space, which is noisy and not necessarily all-encompassing. We then formally consider the labelers as instantiations of a generative model subject to different conditions, the original labeler can be recovered. Instead of manually determining labels, structured labeling enables the completion of unlabeled datasets with a roughly constant number of programmed heuristics. Despite the success of Bach et al. [2017] in heuristic based label collection, we were unable to agree on objective heuristic parititonings; that is, each individual heuristic proposed was itself not self-consistent.

EigenTurk In answer to the problem of consistency within MOCs, a strong enough partitioning of dialogue into subproblems is one that on average, appears to be differentiating to actual conversationalists. In other words, although there aren't objective heuristics, separating turns of dialogue that humans perceive as drastically contained in different MOCs is sufficient for simplifying the learning task for a dialogue module in each partition. Motivated by the foregoing logic, we built a Mechanical Turk task called EigenTurk for annotating randomly selected subsets of conversations from our aggregated corpus.

The Mechanical Turk task is as follows. Users are presented with N lines of dialogue from a randomly selected corpus as in Figure 2.2. They must navigate between each utterance as separated by the aforementioned NLTK processing step. At each utterance a context menu appears on the side of the particular dialogue which contains a hierarchical selection scheme for the particular MOC annotation for the utterance. After the annotation is selected, the user moves to the next utterence and continues until the entire document is completed. In general, the user only sees a small subset of a full dialogue so as to make the labeling robust in expectation.

3 System Design

3.1 MOC Models

Our system chose to create separate models for each of our Modes of Conversation (MOC). To fit in the paradigm of our MOC division, we decided to make a separate model for each individual MOC. For some of the above MOCs there are corresponding models whose state of the art performance is acceptable. However, many other modes of conversation remain open areas. To make use of each mode of conversation, we realized that we need to use a discriminative model to map from text to mode of conversation.

3.2 Discriminator

Formally, we built the discriminator to solve the following two problems. Let $\mathcal{M}$ be a set of conversational modes and $D_t = ((s_i, x_i))_{i=1}^t$ be a list of dialogue up to time step t where s_i is the speaker and x_i is the string (utterance) spoken at time i.

- Given a new utterance $u = (s', x')$, we would like to find a map $\mathfrak{d} : (D_t, u) \mapsto m \in \mathcal{M}$; that is, we would like to classify utterances as MOCs.
- Furthermore, we would like to learn the conditional probability $P(m|(D_t, u))$; that is we would like to establish a posterior on the mode of conversation for time $t + 1$ given a conversation up to time t. Then we call $\mathfrak{s} : (D_t, u) \mapsto m \sim P(m|(D_t, u))$ the selector for the next most probable mode.

Alternatively: Suppose each conversationalist is an agent in an environment E^{MOC} where their action space is $\mathcal{M}$. We would like to learn the action-value function, $Q^\pi((D_t, u), m))$, for one agent in the conversation, π, and define our selector $\mathfrak{s} : (D_t, u) \mapsto \arg\max_m Q^\pi((D_t, u), m)$. Effectively, select the action which most maximizes the reward of the conversationalist, and in this case, we avoid dealing with a multi-modal selector, or an underconstrained problem. (See inverse reinforcement learning for conversation).

Task (1) is useful as we would like to label conversation data according to its MOC and then train each module for that MOC only on data specifically labeled as that MOC. Furthermore, once task (1) is complete, Task (2) yields a selector which essentially selects the next best/probable MOC to use after a dialogue.

3.2.1 Design

Featurization . Given some raw utterance text x we would like to find a map $\phi : x \mapsto \phi(x) \in \mathbb{R}^n$ for use in some discrimination model. Therefore we propose the following featurizations in order of increasing complexity:

- Standard word2vec. Let $\phi(x) = \sum_{i=1}^{\text{length}(x)} \text{word2vec}(x_i)$. Then $\phi(x) \in \mathbb{R}^{\sim 100}$.
- Convolved word2vec. In this case we can vectorize the text by first concatenating sequential tuples of word2vec embedded words; that is let $c_t = [\text{word2vec}(x_t^{(1)}), \text{word2vec}(x_t^{(2)}), \text{word2vec}(x_t^{(3)})]$ and then using some weight matrix W map $c_t \to h \in \mathbb{R}^{\sim 100}$.
- Composed recurrent utterance embedding. For an utterance $u = (s, x)$, we consider a neural approach as follows. Generate a text vector for each word, $\tau = [\text{word2vec}(x^{(1)}), \cdots, \text{word2vec}(x^{(n)})]$ where $x^{(k)}$ is the k^{th} word in x. Then apply a recurrent model to τ at each embedded word, yielding a hidden state vector $h^{(k)}$. Then let $\phi(x) = h^{(n)}$; that is, let the embedding in this case be the last hidden state vector of the recurrent model applied to the sequence τ.

Model. With a reasonable embedding for utterances given we give three different model schemas with increasing complexity. First and foremost, the simplest model is context free. The model essentially takes some embedding $\phi(x)$ and maps it to a MOC label. In this case x is the particular dialogue utterance to be classified, and no other information is therefore accounted for.

For the second iteration of the discriminator, we use the entire dialogue D_t and the current utterance u to produce an MOC on u. In this scheme, we apply the embedding to every utterance in the dialogue history D_t and produce a vectorized history, $D_t^\phi = ((s_i, \phi(x_i)))_{i=1}^t$. Then for each utterance (embedded text-speaker pair) in D_t^ϕ, we apply a recurrent model to the embedded text and the one-hot vector of the speaker. The recurrent model is bifurcated for two types of inputs; the first being these hidden content vectors, and then the second being the embedded current utterance $u^\phi = (s, \phi(x))$. This model trains end to end and yields a prediction on the MOC.

The final instantiation of the model is a hybridization of the dynamic memory networks proposed by Kumar et al. [2015]. Replacing the question module with the current utterance embedding and the answer module with a MOC prediction module.

The Selector. The models previously specified for the discriminator may be adapted to perform the task of the selector $\mathfrak{s}$, by augmenting the featurization of each dialogue piece with the MOC under which it was classified, and then requiring that the model predict the most likely next MOC for a future piece of dialogue instead of having some "current utterance" which it needs to classify. Formally this is as follows: Given some $D_t^\phi = ((s_i, \phi(x_i)), \mu_i)_{i=1}^t$ we learn a model from any of the three aforementioned schemes so that the model predicts μ_{t+1} without a prior on (s_{t+1}, x_{t+1}).

3.3 Generators

3.3.1 Baseline

We began each mode of conversation with a baseline model using AIML, a markup language that matches an input piece of dialogue with a predefined response. AIML files are loaded into an interpreter. Whenever the interpreter is fed a dialogue query, the interpreter finds a matching two-turn dialogue where the first turn matches the query with a set of pre-defined responses. Special templating is set up to store nouns or verbs from the query and these stored values can be invoked through the AIML. The model's rigid restriction to mostly two-turn conversations, with the exception of conceptual templating that was often haphazard, this model managed to provide very interesting conversations upon our internal tests with the model.

3.3.2 Extended Memory Networks

Background Having decomposed the dialogue history $D_t = (s_i, x_i)_{i=1}^t$ into an appropriate MOC $m \in \mathcal{M}$, we now pass the dialogue through a model trained on conversation data restricted to the MOC m. The approach we take to architect and train these models is as follows. Recurrent models, in particular LSTM variants, have proven adept at learning the latent linguistic structures necessary to perform benchmark tasks in conversational AI, such as language models. Though such models consistently outperform other approaches, such as n-gram based techniques, when applied to conversational tasks, the networks struggle to track context and long-term logical dependencies. A stronger approach is to augment the recurrent model with a memory matrix, inspired by neural turing machines Graves et al. [2014].

This is the approach taken in Sukhbaatar et al. [2015], which is characterized by *memory hop* layers. Given a set of input sentences $\{x_i\}$ and a query sentence q, represented as a sequence of one-hot word vectors, a memory hop layer passes each x_i through an embedding layer which encodes the sentences into a vector m_i, and stores each in a memory matrix. This is followed by taking q, embedding it in a different space as a vector u, and computing the cosine similarity between u each m_i. The resulting vector is passed through a softmax with the intent of getting the model to learn a probability vector p over the input words. We combine p with each x_i, embedded in a new space as c_i, by summing them, weighted by their probability of relevance p_i, to obtain a response vector o. The layer's output is a simple combination of the response vector and the embedded query, $o + u$. By stacking multiple memory hop layers on top of each other and passing the final output through a weight matrix and softmax, this model is shown to perform well in synthetic question-answer experiments.

Modified Approach Memory networks were initially designed to target synthetic question-answer experiments, wherein a sequence of factual statements is stored in memory, a question is passed as the query, and a single word answer is outputted. The model can be adapted to perform well in word-level language modeling tasks by passing in a window of previous sentences as a sentence context to be stored in memory in order to predict the next word, with the query sentence fixed to an arbitrary constant vector. The model performs at state-of-the-art level on both tasks.

Our approach is to first extend the memory network from learning a probability distribution over words in the vocabulary to learning one over embedded sentences instead. Given N sentences, we embed the first $N - 2$ into memory as the sentence context and embed the second to last sentence in as the query, with the intention of predicting the final sentence. We use a position encoding scheme to embed sentences, by embedding each word in the sentence with the same embedding and summing in a way that weights each word according to its position in the sentence. To appropriately modify the memory network for this task, we pass the output of the final memory

hop layer through a linear transformation before passing through a dropout-regularized LSTM recurrent network that transforms the embedded output back into a sequence of one-hot vectors. The RNN stacked on top of the final memory hop layer is essentially meant to work with the underlying memory network to learn how to invert the embedding as necessary for supervised training.

Finally, we use a custom loss function designed with the following considerations in mind. The default sequence-to-sequence loss on one-hot vectors commonly used is inadequate for learning a sentence-level language model due to its artificially high sensitivity to frameshift errors. Further, most loss functions that act on sequences of one-hot vectors act through some similarity measure on the tokens composing the sentence, as opposed to a loss function which tries to measure the distance between two sentences in a way that accounts for the sentences' respective underlying meanings instead of just their semantic composition. To this end, we compute the loss as the sum of two loss functions - one which computes the average cosine distance between corresponding words in the sentences, after each word has been re-embedded (either in the same space as the initial memory embedding or using a pre-trained embedding such as word2vec), and another which computes the cross entropy loss on sentence embeddings. We train the entire model end-to-end.

Transfer Learning for Conversation Generation Having trained a model to predict the sentence following a window of previous sentences, we now apply transfer learning techniques to shift the focus of the model from prediction to generation, by re-training a trained model on conversation data. We pass the dialogue D_{t-1} in as the sentence context and the immediately previous response x_t as the query. By forcing the model to predict the next response, it learns to generate meaningful responses to queries through a methodology that accounts for the contextual history of the conversation up to that point. Transfer learning is an instrumental component of the training process, simply due to the immense disparity in the amount of available data for learning a language model - for which raw text is suitable - and that information available for conversational AI - which is highly unstructured, diverse, and comparatively rare. We theoretically justify this approach through the intuition that having been trained to predict sentences on large amounts of structured text, the model is forced to learn to represent sentences in a continuous, meaningful way that captures some notion of the sentence's underlying linguistic meaning and relationship to neighboring lexemes and linguistic constructs. By fixing the embedding matrices as constant, the network is now able to learn to generate responses to queries in a supervised way that takes into account, a priori, both the meaning of sentences in the conversation history as well as their relationship to each other.

4 Infrastructure for Scalable Conversational AI

We approached the engineering side of our Alexa Prize bot with three basic tenets in mind: modularity, scalability, and reliability. We call the system Eigen Brain. The modularity of Eigen Brain allowed us to easily achieve both scalability and reliability. We also decided to ensure that Eigen Brain was a stateless application, storing its state only in a database, rather than in memory, which allows us to make use of server-less architectures such as AWS Lambda. This means we can rely on Lambda's robust scaling system rather than one one that we build - significantly reducing the number of potential uncaught bugs.

4.1 Modules

Eigen Brain is made up of four kinds of modules: generator, discriminator, state management and logging modules. The general arrangement of these modules is described in Figure 4.2. Generator modules are responsible for the actual generation of conversational text. Typically, each generator module is designed to respond to one MOC, however, the system is general and doesn't depend on this. Discriminator modules determine what MOC the conversation is in, and therefore what response type is appropriate. Depending on the output of discriminator modules, different generator modules will be run. The state management modules are responsible for extracting state information from incoming dialogue, so that generator modules can utilize this information in crafting responses. As an example, one state management module might be responsible for determining whether there is a swear word in the text. If there is, then it would add a flag to the state and the generators would recognize this flag and respond appropriately. Finally, the logging modules are responsible for storing

any information that would be useful in analyzing the system at a later date or determining the origin of a poor response or bug.

4.2 Architecture

Because of the modularity of Eigen Brain, there is ample flexibility in the design of the infrastructure that Eigen Brain runs on. For example, if one particular module is computationally expensive it can be given its own server, possibly with GPUs to accelerate its computation, or multiple less expensive modules can all be run on the same server. In the end, we decided upon having one AWS Lambda function to run all the discriminator and state management modules, a central MongoDB database to store the state that can be accessed by any of the Lambda functions or servers, and individual dedicated servers and AWS Lambda functions for each generator module. This enables scalability and reliability because any individual module can easily be scaled up separately from the entire system, and if any one module goes down, so long as at least one module of its type remains, the system will still stay online. For example, if one generator module goes down, the system will continue to work, it will just produce less diverse responses. Additionally, this architecture reduces latency because each generator module can be queried asynchronously and then the fastest response can be redirected to the user without needing to wait for all the generators to finish crafting a response.

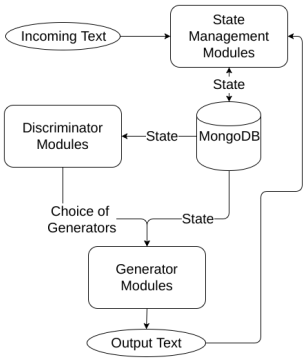

Figure 3: System Architecture Illustrated.

References

Stephen H. Bach, Bryan Dawei He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. *CoRR*, abs/1703.00854, 2017. URL `http://arxiv.org/abs/1703.00854`.

Jeremy H. Clear. The digital word. In George P. Landow and Paul Delany, editors, *The Digital Word*, chapter The British National Corpus, pages 163–187. MIT Press, Cambridge, MA, USA, 1993. ISBN 0-262-12176-x. URL `http://dl.acm.org/citation.cfm?id=166403.166418`.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*, pages 699–708, 2012.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *CoRR*, abs/1611.08669, 2016. URL `http://arxiv.org/abs/1611.08669`.

John W. Du Bois, Wallace L. Chafe, Charles Meyer, Sandra A. Thompson, Robert Englebretson, and Nii Martey. Santa barbara corpus of spoken american english, parts 1-4. Philadelphia: Linguistic Data Consortium., 2000-2005.

John Godfrey and Edward Holliman. Switchboard-1 release 2 ldc97s62. Philadelphia: Linguistic Data Consortium, 1993.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. URL http://arxiv.org/abs/1704.03477.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. URL http://arxiv.org/abs/1412.5567.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015. URL http://arxiv.org/abs/1506.07285.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *CoRR*, abs/1506.08909, 2015. URL http://arxiv.org/abs/1506.08909.

Elisa D. Mekler, Florian Brühlmann, Klaus Opwis, and Alexandre N. Tuch. Do points, levels and leaderboards harm intrinsic motivation?: An empirical analysis of common gamification elements. In *Proceedings of the First International Conference on Gameful Design, Research, and Applications*, Gamification '13, pages 66–73, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2815-9. doi: 10.1145/2583008.2583017. URL http://doi.acm.org/10.1145/2583008.2583017.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. Effects of age and gender on blogging., 01 2006.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. Multiresolution recurrent neural networks: An application to dialogue response generation. *CoRR*, abs/1606.00776, 2016a. URL http://arxiv.org/abs/1606.00776.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. 2016b.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015. URL http://arxiv.org/abs/1503.08895.

Marilyn Walker, Pranav Anand, Jean Fox Tree, Rob Abbott, and Joseph King. A corpus for research on deliberation and debate. *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, 01 2012.

Kyle W Willett, Melanie A Galloway, Steven P Bamford, Chris J Lintott, Karen L Masters, Claudia Scarlata, BD Simmons, Melanie Beck, Carolin N Cardamone, Edmond Cheung, et al. Galaxy zoo: morphological classifications for 120 000 galaxies in hst legacy imaging. *Monthly Notices of the Royal Astronomical Society*, 464(4):4176–4203, 2016.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.