Snowbot: An empirical study of building chatbot using seq2seq model with different machine learning framework

Pinglei Guo piguo@ucsc.edu
Yusi Xiang yxiang12@ucsc.edu
Yunzheng Zhang yzhan300@ucsc.edu
Weiting Zhan wzhan83@ucsc.edu

Abstract

Chatbot is a growing topic, we built a open domain generative chatbot using seq2seq model with different machine learning framework (Tensorflow, MXNet). Our result show although seq2seq is a successful method in neural machine translation, use it solely on single turn chatbot yield pretty unsatisfactory result. Also existing free dialog corpus lacks both quality and quantity. Our conclusion it's hard to build a useful open domain generative bot using state of art technology.

1 Introduction

1.1 Chatbot

Chatbot can be generally divided into two types, open domain and close domain. Former can answer a wide range of questions (though the answer itself could be very general with syntax and semantic error), latter can solve particular problems from user, and user is not expecting a close domain bot to chat freely as well. Based on how response is generated, it can be divided to retrieval and generative, where answer is picked from clustered responses [5] or generated on the fly. In theory, open domain chatbot has more potential[1], and is harder to build, in practice, most chatbot that gets work done is close domain, even those that seems to be open domain (Siri, Cortana) are not implemented end to end, heuristic rules, entity recognition, knowledge base, clustered response are inject into multiple stages in the pipeline to produce good result.

At first (during proposal) we believe a close domain retrieval chatbot is the most easy one in the total four combinations based on a blog post [4], Then we realize that this is the most easy one in theory but the hardest one in practice, the other corner, open domain generative is actually easiest because it's end to end, just two RNN is enough. Thus we decided to build a open domain generative chatbot in the limited time given.

1.2 Machine Learning Frameworks

Machine learning framework is now the competing spot for tech giants (Table 1), besides they are using ML excessively in their own products, they want developers to use their framework and run it on their cloud platform, (i.e. Tensorflow on Google Cloud, MXNet on AWS, CNTK on Azure). Although there were many open source frameworks started by individual and research organizations, most of them are deprecated in favor of those backed by companies (i.e. Theano vs. Tensorflow)[2]

[1] if the domain is really open enough, generate random text might be the best algorithm

[2] https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ

Table 1: Popular Machine learning Frameworks

Name	Star	Company	First Release
Tensorflow	81697	Google	2015-11-08
Keras	22872	N/A	2015-01-13
Caffe	21739	N/A	2014-03-19
CNTK	13366	Microsoft	2016-01-22
MXNet	12392	Amazon	2015-12-09
PyTorch	10142	Facebook	2016-08-31
Torch	7537	N/A	2015?

Tensorflow [1] and MXNet [2] are two widely used frameworks, while tensorflow is mostly declarative, mxnet allows more imperative programming style. A simple example is in order to debug the value of a matrix in tensorflow you need to write an op using `tf.Print` and put it inside compute graph, while in mxnet you can just use `print` like it's a normal python variable, though in fact it might resides on GPU or other machines. Declarative style is not very expressive, but make optimization on the framework side easier, mxnet chose to detect patterns and only optimize critical path to gain flexibility while retain performance. However, when it comes to most software engineers, the design and low-level API does not matter much, community support and a wide range of up to date model is more important, and tensorflow is the obvious winner. For researchers, mxnet might be a better choice, since it has thinner wrapper and stricter requirement.

We chose to use two higher level seq2seq frameworks, sockeye (based on mxnet)[3] and OpenNMT-tf[4] in the experiments, we did try to write a naive one based on newer seq2seq API in tensorflow, however since both sockeye and OpenNMT-tf use state of art models and yield unsatisfactory result, we abandoned it in the middle [5].

The rest of the report is organized as following, Section 2 describes the basic form of the seq2seq model we are using. Section 3 shows how we process our dataset, and the problem in it. Implementation detail is listed in section 4. Our result is shown in section 5. Section 6 concludes the report and **workload distribution**.

2 Model

We use seq2seq model, which is widely used in neural machine translation [9] and can be applied to single turn dialog system (QA) as well. Its basic structure is two recurrent neural networks (RNN) as shown in figure 1. The cell used in RNN is LSTM (long term short memory), so we can learn implicit relation ship between words from data without explicit pre-processing. seq2seq is one of many variants of RNN + LSTM. Based on the domain and length of input, RNN + LSTM can be used for sequence classification, text generation, translation, dialog system etc. as shown in table 2

Table 2: Variants of RNN + LSTM

Type	Input	Output	Example
seq2one (classification)	sentence	class	Sentiment analysis
seq2seq (text generation)	sentence	same sentence	Shakespeare style text
seq2seq (translation)	language A	language B	NMT A -> B
seq2seq (single turn QA)	question	answer	chatbot
seq2seq (multi turn QA)	conversation context	answer	chatbot

[3] `https://github.com/awslabs/sockeye`

[4] `https://github.com/OpenNMT/OpenNMT-tf`

[5] `https://github.com/at15/snowbot/pull/23`

For many to many in RNN + LSTM (seq2seq), simply swap the input and output data, you get a different application for free (Table 3) [6]. So we can use neural machine translation framework directly to build our chatbot.

Table 3: Input and Output of different seq2seq application

Type	Train input	Train output	Test input	Test output
text generation	You are my foe	You are my foe	You are	my <unk>
translation	你好	Hello	吃了么	Good morning
single turn QA	Any idea	I don't know	How's the weather	I don't know
multi turn QA	Got it?; No; Why?	I don't know	Hi; Hey; What's up?	I don't know

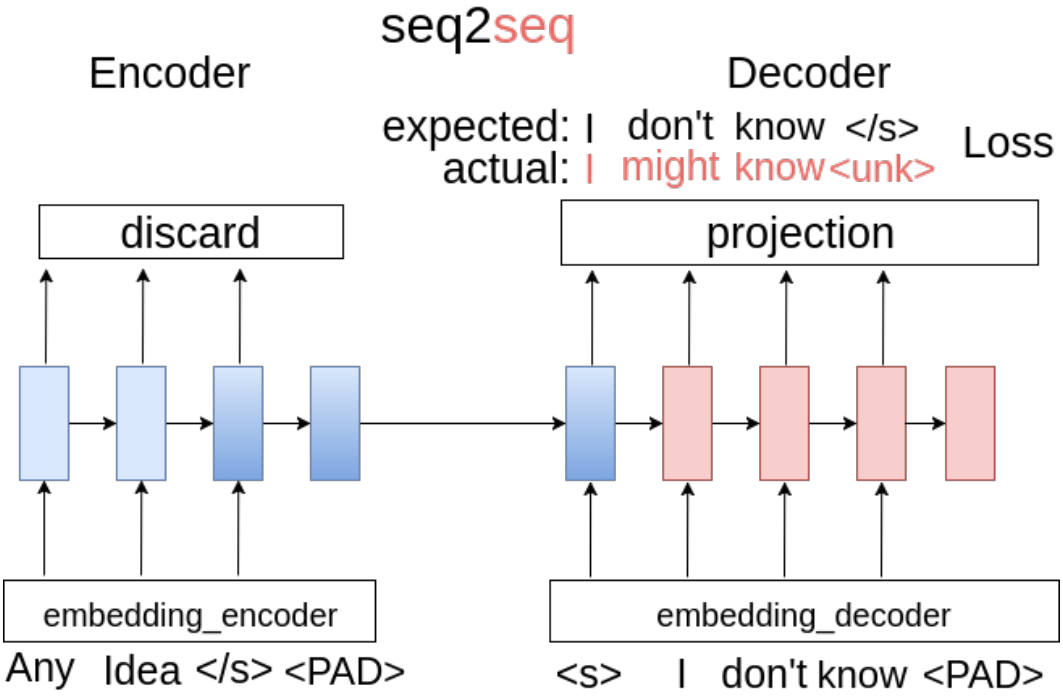

Figure 1: Sequence to Sequence model for single turn Dialog

From figure 1 we can see the main idea of seq2seq is to encode input words sequence into a vector, and decode this vector until reaches max step or EOS (`</s>`). On the encoder side, the output and hidden state are all discarded except the last hidden state, which certainly lose some important information, and leads to optimization like attention [8].

On the decoder side, train and test (infer) is different. The first word is `<s>` instead of real word in both case, but in training, the word feed into each step is from original data while in testing, the word feed into each step is based on the output of previous step. The output from projection layer (after softmax) is a vector indicating probability of each word in vocabulary. In training, we calculate loss use cross entropy, note that due to the length of sequences in a mini batch may not be the same, we use padding (`<PAD>`) to form a matrix and we need to mask those padded part out when calculating loss. In test, we can pick the word with max probability as word for this step, this is the greedy way, but we may not pick the overall best sentence if we consider each word independently. A better way is to use Beam Search, which is keep top-K words in each step, and take the path of top-1 as the final output.

In sockeye and OpenNMT-tf, both attention and beam search are enabled by default, also there can be more than one layer LSTM or combination of cells (bi-direction), in original

[6] not real data, for demonstration only

NMT paper they say feed input in reverse order increase performance [9], and with bi-direction, a sentence is feed into encoder in two directions.

3 Dataset

3.1 Handling I don't know in Cornell Movie Dialog Corpus

We start with Cornell Movie Dialog Corpus [3] because it is used by many publication and blog posts for single turn dialog system. It contains 83097 conversations, and can be converted to 221282 single turn QA, total file size is 22MB.

However, our first run using sockeye shows `I don't know` is the most popular response (Table 4 left), which leads to a very simple questions. How many `I don't know` do we have in training data, does the bot learn to know what it really knows, or there are just too many `I don't know` in training data.

So we add counting and removing `I don't know` when processing text. There are 5114 response is just `I don't know`, which is 2.3%., excluding those with `I don't know` as prefix. Considering the corpus is from movie with varies backgrounds, 2.3% is a relative high percentile. After removing I don't know, the responses (Table 4 right) become very general questions like `what do you mean`.

Table 4: Cornell corpus before and after filtering out I don't know

Raw text without processing	After filtering out I don't know
Q: hi	Q: hi
A: I don't want to talk to you.	A: hi.
Q: do you know	Q: do you know
A: I don't know.	A: i don't think so.
Q: you know what	Q: you know what
A: I don't know.	A: what do you want?
Q: oh say something other than I don't know	Q: oh say something other than i don't know
A: What are you talking about?	A: what do you want me to do?
Q: so this is all you got	Q: so this is all you got
A: I don't want to talk about it.	A: i don't know what it is.
Q: do you like cat	Q: do you like cat
A: I don't know.	A: what do you mean?
Q: what do you know	Q: what do you know
A: I don't know.	A: what do you mean?
Q: where can I find the park	Q: where can i find the park
A: I don't know.	A: i don't want to know what you want.

3.2 Twitter Chat log with emoji

Since Cornell Movie Dialog Corpus is quite dramatic, we decided to try more daily style dialog corpus. The data is from Marsan-Ma/chart_corpus[7], he wrote a crawler to scrape twitter.[8] It contains 377265 QA and there is just 6 pure `I don't know`, total size is 51MB.

One interesting to note is the data contains emoji [9], and we didn't remove it when pre-processing. We could just keep all the ASCII, but we thought it would be interesting to see if emoji would show up in generated response, and they did, as show in Table 5

[7] `https://github.com/Marsan-Ma/chat_corpus`

[8] As for now, using Twitter API requires enterprise plan and share data on github has legal issues

[9] we use :heart: instead of real emoji in report due to there is no such font for latex

Table 5: Cornell Movie Dialog vs Twitter Chat

Cornell Movie Dialog Corpus	Twitter Chat
Q: thank you	Q: thank you
A: I don't want to see you.	A: thank you so much :heart:.
Q: Thank you	Q: Thank you
A: yeah.	A: thank you so much for the <unk>

4 Implementation

4.1 Data Pipeline

Because we decide to use different frameworks's with different dataset, a lot work is done to provide a unified interface for processing the data and feed into the graph. Each framework has their own dataset wrapper and special data format (i.e. `TfRecord`). Instead of writing ad-hoc code to adapt to different code and dataset, we generalize this procedure in package `snowbot.corpus`[10]. So adding a new dataset just need to inherit the class and implement some dataset specific logic like handling invalid input. It has a command line interface as shown in figure 2. The total implementation of the data pipeline has around 1,000 lines of python code, it's not very efficient now, we plan to use C extension and allow short path between phases instead of storing intermediate result on disk.

```
⇒  python -m bin.corpus --help
Usage: corpus.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  convert       Convert raw data to format like csv
  download      Download and extract corpus to ./data
  download_file  Download file to folder, like wget/curl
  extract_file   Extract .zip/tar.gz to folder, like tar w/o...
  gen_qa        generate one turn dialog (QA)
  gen_vocab     generate vocabulary
  list          Show known Corpus that can be downloaded
  split         split qa into train and test set
```

Figure 2: Data Pipeline command line interface

4.2 Text processing

In contrast with previous assignments, we removed a lot of text processing when building the chatbot, just simply split word by space and ignore invalid codec that can crash the program. For instance, stop words are not removed, because in a normal chat, we are expecting words like `I`, `a`. Also we find converting words to lower case does not make much difference (see section 5) because they will have similar word vector given enough data and training time. Special characters are also not removed, so certain dataset can keep their own feature, like emoji is widely used in twitter (see section 3.2).

However, we added an extra step of removing high frequency general dialogs from corpus like `I don't know`, which gives use a better result, but the alternatives are still quite general. Another addition formatting dataset into a standard format of question and answer. All of the text processing are in our data pipeline tool (see section 4.1) with around 600 lines of python code.

[10] `https://github.com/at15/snowbot/tree/master/snowbot/corpus`

4.3 Seq2Seq Frameworks

In our experiments we use existing seq2seq frameworks, there are plenty of them, as shown in Table 6. We chose OpenNMT-tf and sockeye because we are interested in the rumor that MXNet is more efficient on using GPU memory than Tensorflow.

Table 6: Seq2Seq Frameworks

Name	Framework	GitHub	Star
OpenNMT-py	PyTorch	OpenNMT/OpenNMT-py	773
OpenNMT	Torch	OpenNMT/OpenNMT	1461
OpenNMT-tf	Tensorflow	OpenNMT/OpenNMT-tf	145
nmt	Tensorflow	tensorflow/nmt	2094
seq2seq	Tensorflow	google/seq2seq	3019
tensor2tensor	Tensorflow	tensorflow/tensor2tensor	2975
Sockeye	MXNet	awslabs/sockeye	290

Although most of them are built for neural machine translation, we can use them for chatbot directly by swapping the input and output file as show in Table 3.

4.3.1 Sockeye (MXNet)

Sockeye only requires two text file, it was expecting two different language, but we replace it with questions and answers in same language. It will generate the vocabulary on the fly, it simply splits sentence into tokens by space, no conversion like remove special token is performed. (But we did the pre-processing and joined the processed text with space.) It supports attention and beam search.

4.3.2 OpenNMT-tf (Tensorflow)

OpenNMT-tf use a YAML file as config, which is similar across all OpenNMT implementation. It also supports attention and beam search, but its default configuration is not very practical. It asks you to set train step instead of epoch, and seems don't support auto stop when there is no more improvement after k epoch.

5 Evaluation

We run our experiment on a normal desktop with a NVIDIA GTX 1080 8GB. We use 2 layer LSTM with 256 hidden units, (took around 5GB RAM and has 66157396 parameters), vocabulary size is 50000 (covers 90%+). Attention is used and beam size is set to 5 for beam search (there is no randomness injected, so the response is quite deterministic). Normally it took 1h (around 7 epoch) for sockeye to reach the best result, it will continue 3 epoch before stop training. For OpenNMT-tf, the default config would take 83.7h to finish training (1000000 steps), so we force it to stop after 1h.

One thing to note is there is no hard metrics for evaluating a chatbot [6]. We use perplexity like NMT, to represent how close we approach when training. For Cornell dataset, the best perplexity in training on evaluation set is 90, for Twitter dataset, the best perplexity is 135. However, the most popular way for testing a dialog system now is to employ human to give judgement (i.e. is this a bot or a human), which is quite expensive, so we skipped it and only tested it among team members.

5.1 Word embedding

We didn't initialize our embedding matrix with pre-trained value from word2vec or GloVe, and let they be part of the parameters and trained along. Although our data is not large, but it's big enough to capture the similarity of words, so even without careful text processing (section 4.2), our eventual result won't suffer much from that, because words like `love` and

`Love` does not have very large distance as shown in figure 3 [11]. We did normalize text in some runs, but since it's hard to evaluate automatically, we find little difference for the limited input we have tried.

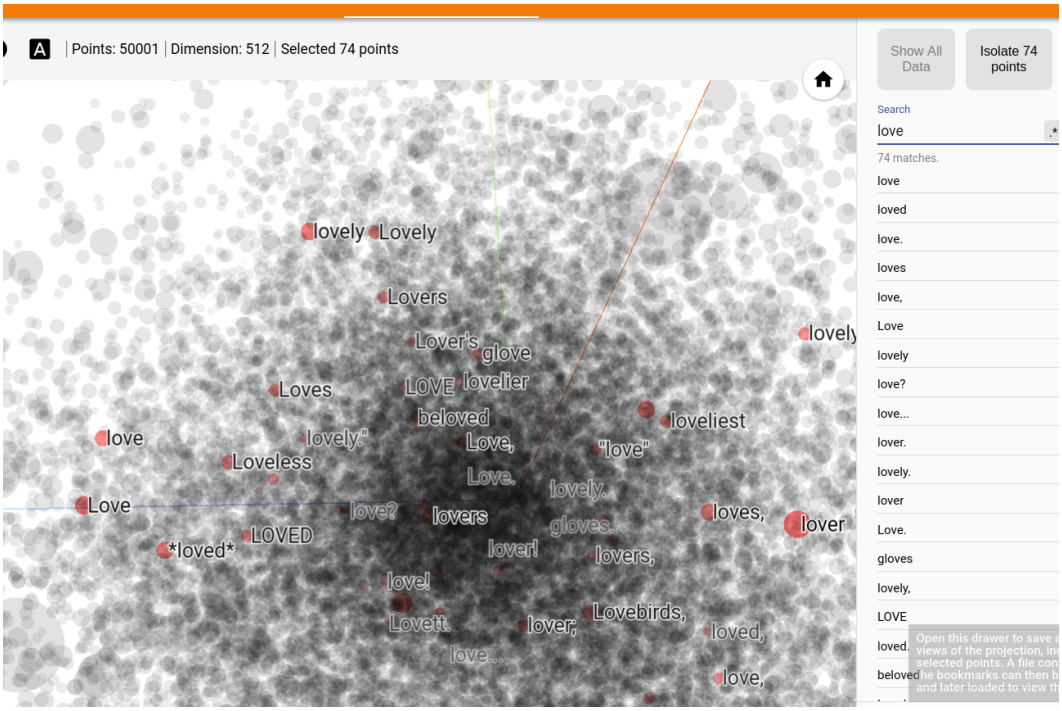

Figure 3: Word embedding with random initialization

5.2 Entity recognition

Although we didn't implement named entity recognition (which is a core part of industrial strength chatbot). Some dialog can shows our naive end to end framework can capture simple entity like refer to different person `he, she, I` (Table 7). However the grammar can be very wrong i.e. `I don't what I was`, it's likely to be caused by `I was` appers much more than `I am` in training data.

5.3 Generic Response

There are a lot generic responses in the chats we have listed (Table 4, 5, 7). It's a common problem if look at all the chatbots using seq2seq only. Attention is said to be able to improve this, but we didn't see much difference when attention is applied. We believe this can't be solved easily if we stick to a simple end to end framework like seq2seq, extra knowledge must be introduced, like named entity recognition. When training data is relative large, the model will fit to the most common dialogs, which falls to `I don't know, what do you mean`. Some people chose to train the already trained model with very small dataset to generate more vivid dialog, which we believe is just a more advanced trick of using hard coded rules to shrink the domain of response and is not a general solution.

Also, we believe the generic response also have a lot to do with our dataset. The Cornell Movie Dialog Corpus we are using is quite dramatic, it's script for movie after all. It would be better to have corpus that covers people's daily life. Ubuntu Corpus [7] might be a good choice, it is very technical but more closer to daily life. Twitter no longer allows people to scrap and share data, people doing that has legal issues. There are people sharing Reddit data (1TB) but we haven't tried that yet.

[11] you can also use tensorboard for mxnet based application `https://github.com/dmlc/tensorboard`

Table 7: Simple entity recognition

Cornell Movie Dialog	Twitter Chat
Q: what do you think about him A: what do you mean? Q: where is he A: i don't know what he is. Q: where is she A: she said she was a <unk> Q: she said what? A: she said she was a <unk> Q: he said what? A: he said he was a <unk> Q: i said what? A: what do you want?	Q: where is he A: i think he is a <unk> Q: where is she A: i don't know what she was in the <unk> Q: where is him A: i think he is a <unk> Q: where is her A: i don't know what she was in the <unk> Q: where am i A: i don't know what i was in the <unk> Q: what is he doing A: i think he is a <unk> Q: what is she doing A: i think she is a <unk> Q: what am i doing A: i don't know what you want to do with the <unk>

6 Conclusion

We built an open domain generative chatbot using seq2seq model with Tensorflow and MXNet respectively. The bot is able to capture simple entity, but most responses are pretty generic. End to end model using seq2seq is easy to implement, but using it alone can't yield good result. In the future we plan to try entity recognition and larger dataset with distributed training, and might switch to retrieval based chatbot.

The workload in the project is distributed as following

- Pinglei Guo: Data Pipeline, Part of Tensorflow of MXNet based implementation
- Yusi Xiang: Twitter Data, Part of MXNet based implementation
- Yunzheng Zhang: Cornell Data, Part of Tensorflow based implementation
- Weiting Zhan: Twitter Data, Test, Report

The authors like to thank Tianyi Luo and Haiyu Yang for advice on the seq2seq model and organizing python codebase.

References

[1] Mart í n Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Tianqi Chen et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". In: *arXiv preprint arXiv:1512.01274* (2015).

[3] 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.

[4] *Deep Learning for Chatbots, Part 1 - Introduction*.

[5] Anjuli Kannan et al. "Smart reply: Automated response suggestion for email". In: *arXiv preprint arXiv:1606.04870* (2016).

[6] Chia-Wei Liu et al. "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation". In: *arXiv preprint arXiv:1603.08023* (2016).

[7] Ryan Lowe et al. "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems". In: *arXiv preprint arXiv:1506.08909* (2015).

[8] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective approaches to attention-based neural machine translation”. In: *arXiv preprint arXiv:1508.04025* (2015).

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.