Project Task: Chatbot

Silje Christensen
NTNU
siljechristensen92@gmail.com

Simen Johnsrud
NTNU
simen.johnsrud92@gmail.com

Abstract

TBA
First version.

1 Introduction

Conversational agents have been a hot topic of research area the past years. Our goal is to look at different datasets, and to explore different models in order to find the best one. Our long term goal is then to further explore variants of this model.

With the increasing amount of large conversational data available, we can feed a chatbot with more knowledge and get smarter systems. Several papers has explored how the best versions of certain models can be implemented, yet, we are still not there.

2 Motivation

Even though conversational agents is a hot topic of research, you do not encounter many intelligent and useful chatbots in your everyday life. We are just not there yet. But imagine all the possibilities a general and intelligent chatbot could give us. We believe that exploring new models used for chatbots and improve the current state of art will make the foundation for new innovative solutions that we, in the future, will take for granted.

With this as a baseline, we want to explore the open source Ubuntu Dialogue Corpus and hopefully a private customer support dataset from Telenor, and further implement both a retrieval based and a generative based architecture for our RNN based chatbot. In both cases, the datasets are closed-domain. We aim to get a better understanding of the two main categories within system dialogue development, in order to make a better version of the current state of art. Further, we will look at the possibility to combine them, or to improve the best candidate to make a useful chatbot.

3 RNN

A vanilla RNN implementation is a recurrent network that uses its own calculations as inputs. The network is more complex than a ordinary neural network, but the architecture is really much the same. For training purposes, the most common way is to use Stochastic Gradient Descent (SGD), but since the parameters are shared by all time steps in the network we need to change how we calculate the gradients for SGD. Backpropagation Through Time (BPTT) is one of the popular solutions. By doing this, the network is able to have some conditions between the input, which could be beneficial in the means of conversations. However, a vanilla RNN is unable to learn dependencies

between words that are several steps away, not to mention repliques away. With this as a guideline, Long Short-Term Memory (LSTM) has gained better results as this kind of cells are able to create dependencies between words further away from each other then the original RNN. Vanishing gradients have also been a big problem, and a LSTM architecture is one improvement to this.

4 State of the Art

Different papers of how we can design a better conversational agent has been proposed during the recent years. The authors have different perspective and thoughts when it comes to modelling the best conversational agent. We differ between two main categories of conversational system models: retrieval based and generative based. The first architecture uses a repository of predefined responses and the latter will generate new sentences, which it may not have encountered before. There are pros and cons with both approaches, and both (Shang et al. 2015) and (Lowe et al.) describes why one model is better than the other with different point of view.

Retrieval based methods often rely on manual effort in designing rules or automatic training of the model, and they do not necessarily need a huge amount of data. The major drawback of this model is that it can only respond with sentences it has already seen, which makes it difficult to develop an extendable open domain conversational system. These facts makes (Shang et al. 2015) critical to the model. (Shang et al. 2015) embraces the increasing amount of public conversational data, and focuses on a generative approach using an encoder-decoder architecture. This model is commonly used to solve translation problems (Sutskever et al. 2014, Cho et al. 2014, Bahdanau et al. 2016). Instead of feeding the model with an input and output as English and French, we rather feed it with both replies and responses in English. By digging into the encoder-decoder approach, we can see that the model consists of two different RNNs. An encoder that reads and encodes source sentences into fixed-length vectors and a decoder that outputs a translation from the encoded vector (Sutskever et al. 2014, Cho et al. 2014). (Bahdanau et al. 2016) proposed a new version of the earlier described encoder-decoder model to solve the translation problem. The architecture consists of a bidirectional RNN (BiRNN) as an encoder and a decoder that emulates searching through a source sentence during decoding a translation. In their proposed scheme, they want the annotation of each word to summarize not only the preceding words, but also the following words. A BiRNN consists of forward and backward RNNs. The forward RNN reads the input sequence as it is ordered from the first word to the last word, and calculated a sequence of forward hidden states. The backward RNN reads the sequence in the reverse order, resulting in a sequence of backward hidden states. The results show that their architecture outperforms the conventional encoder-decoder model significantly, regardless of the sentence length and that it is much more robust to the length of a source sentence.

It is important to note that compared to the chatbot challenge, the translation problem is significantly easier to solve, and easier to work with as it has well defined evaluation methods, e.g. BLEU. (Shang et al. 2015) propose a Neural Responding Machine (NRM) and employs a neural encoder-decoder to solve Short Text Conversation (STC). The decoder is a standard RNN language model except that it is conditioned on the context input c (combination of the hidden representation and the attention signal), using GRU-cells. They consider three types of encoding schemes, a global scheme, a local scheme and a hybrid scheme which combines the two. The global scheme uses the final hidden state as the global representation of the sentence. The local scheme uses an attention mechanism that allows the decoder to dynamically select and linearly combine different parts of the input sequence. Both schemes has its pros and cons, and by combining them to a hybrid scheme, they get better results.

There is an obvious advantage to the generative model, but there are several reasons for why the current proposed generative models are not good enough for conversational agents, and though it has decent results for short text conversations, it is not good enough for longer sentences. The grammar may be wrong, and the structure of the sentence may not make sense at all. These problems are solved with the retrieval based model, as the sentences is picked from a fixed set, without grammar errors. Several papers has looked towards this approach (Lowe et al. 2016 +++). They consider a TF-IDF (term frequency-inverse document frequency) approach, as well as an RNN and a LSTM approach. The RNN version consists of one encoder RNN, used to encode the given context, and another RNN which generates a response by using beam-search. The major difference between

(Lowe et al. 2016) and the other mentioned papers is that they are concerned with classification of responses, instead of generation. In addition to the RNN model, they consider the same architecture but changed the hidden units to LSTM units in order to model longer-term dependencies. Their results showes that the LSTM model is significantly better than pure RNN and TF-IDF evaluating with Recall@k.

5 Exploring datasets

During the last years, a lot of qualitative, as well as quantitative, datasets have been made available online. However, to find the perfect dataset for a chatbot is still a challenge. Conversations are often private and hence a seldom resource in order to train a chatbot. Opensubtitles[KILDE], reddit comments[KILDE] and twitter comments[KILDE] have all been used, but has the significant drawback of high noise. The conversations are often just replies or short input to a problem. Chatbots would do their best utility if they were able to answer questions from the user, and hence we would need a dataset that could learn the agent to give meaningful responses.

We are aiming for a closed-domain chatbot, that could serve as a customer support employee which means that we should struggle to find old customer support datasets or a FAQ based resource. The Ubuntu Dialogue Corpus, which is a massive set of 1 million multi-turn dialogues [3], will be a good start for us. The conversations here are of a minimum of 3 turns each, with an average of 8 (DET ER PAPERET SOM HAR GJORT DETTE NB NB). In addition to the Ubuntu Dialogue Corpus, and even more qualitatively, we might be able to use a private customer support dataset given by Telenor. (MER HER NR VI VET HVA SOM SKJER)

6 Preprocessing data

All data comes in a greater or lesser extent of noise. Some of this is hard to optimize, but others such as typos and abbreviations is something we can get rid of and hence, improve the quality of the dataset. Normalizer [8] is a library that normalize, clean and fix text that could help us with this. To tokenize the words, NLTK [9] is another library doing this exactly and as a supplementary service, it can also display a parse tree that can be convenient to us.

After tokenize our words, we need to create a dictionary containing the n most common words, and replacing the others with a $< UNK >$ token. The reason for this is to improve both learning time and the result as knowing more than n words most likely does not make our bot neither more understandable nor useful. In addition we will have to tell our network when a sentence starts and ends by adding the tokens $< GO >$ and $< EOS >$, respectively.

7 Architecture

SKRIV OM ARKITEKTUREN/MODELLEN OG AT VI KOMMER TIL BRUKE LSTM ELLER GRU ELLER NOE SNT

7.1 LSTM

Long short-term memory (LSTM) is an RNN architecture first proposed in 1997 [10] which has been increasingly popular after its origin. The LSTMs are able to learn long term dependencies in a greater extent than before. The layer in each repeating module in the RNN is replaced with a LSTM cell and this is where the magic happens. The most important is the state of the cell. This is like a conveyor belt where the history on it may or may not be affected by new data fed into the module. The change of the state is dependent of the three gates the cell is built up of. First, we have the *forget gate* which decides how much of the old history in the state that should be carried on. All of the gates consists of a sigmoid function outputs values between zero and one. If the forget gate outputs a value of one, it indicates that the all of the history should go through, while a value of zero means that nothing should pass. The second gate, the input gate, decides how much of the new stuff, the input, that should be added to the state. Values close to one means that all of the input should be added. Finally the output gate tells how much of the state that we should output.

7.2 GRU

The resent years, Gated Recurrent Units (GRU), have been very popular as a alternative to the LSTM. The cell consists of only two gates, *reset* and *update* gate. By reducing the number of gates, the model becomes easier to train and hence a performance gain, as well as it is easier to implement. Since it first use in 2004 [6] it has shown good results beside that is performing slightly worse than LSTM, but the reduce in complexity has made it to a popular choice. However, the architecture is still new and not as good researched as LSTMs, which may result in discovering new drawbacks during the next years

7.3 Grid LSTM

TODO TO NEXT THURSDAY! Grid LSTM is a network of LSTM cells arranged in a grid of one or more dimensions [11]. This is analogues to the stacked LSTM, just with the possibility to add cells along the depth dimension too, while a grid LSTM with three or more dimensions equals a multidimensional LSTM, but with the possibility of having cells not just along the depth dimension.

8 How to merge retrieval and genetic models

TODO TO NEXT THURSDAY

9 Ideas

At this point, we got three different ideas for the project:

- The papers read so far compares different variants within one of the main categories. We want to choose a dataset, make one retrieval based and one generative based model to look at the differences. Is it possible to combine the models? Which model should we proceed with on our master thesis?
- Look into grid LSTM. Grid LSTM looks promising in the terms of RNNs. Is it possible to use it to retrieve good results for a chatbot?

As a base of line to all of the ideas, we will probably research new techniques such as attention and GridLSTM in order to improve the current LSTM cell. Both of these are predicted to possibly be the next step for the chatbots. We also want to discover datasets that could help us build a customer support chatbot, which is what we want to end up with independent from which model we land.

10 Challenges

We are facing a lot of challenges, and the list will be increasing as we move on. For now the general challenges is how to build a chatbot model that will work. Then all of the details will follow; Which architecture should we use? How can possible attention improve our results? Will our datasets be good enough? Is it possible to combine a generative and a retrieval model, and if so, what is the most efficient way?

11 Experimental part

12 Exploring datasets

- WMT'14 Translation Dataset (Sequence to Sequence Paper). Open source.
- Telenor customer chat. Private
- Ubuntu Dialogue Corpus. Open source

13 Further work / The road ahead

Start modeling an architecture based on the papers read will be a natural move in this step of the process. We have already read a lot of papers, but there is still more interesting information out there for us to handle, hence will we most likely read some more of these. We will also discover memory networks and look at how attention and GridLSTM possibly can help us to improve our model. This is just two of probably more techniques that could eventually help us gaining better results, which is why we should try to discover more of these.

We need to research on dataset to find one that fits our domain the best. Once obtaining our preferable dataset, we need to preprocess and/or edit the set to fully satisfy our model.

The final goal should be to implement a chatbot that to a greater or lesser extent could serve as a customer support chat. This will be a goal for our master thesis.

14 Conclusion

TBA

15 Acknowledgment

TBA

References

[1] [Bahdanau et al. v7 2016] http://arxiv.org/abs/1409.0473v3

[2] [Dodge et al. v6 2016] http://arxiv.org/pdf/1511.06931v6.pdf

[3] [Lowe et al. v3 2016] https://arxiv.org/pdf/1506.08909.pdf

[4] [Shang et al. v2 2015] https://arxiv.org/pdf/1503.02364.pdf

[5] [Vinyals et al. v3 2015] http://arxiv.org/pdf/1506.05869v3.pdf

[6] [Cho et al. v3 2014] https://arxiv.org/pdf/1406.1078v3.pdf

[7] [Sutskever et al. v3 2014] https://arxiv.org/pdf/1409.3215v3.pdf

[8] [Github] https://github.com/superscriptjs/normalizer

[9] [Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. OReilly Media Inc.] http://www.nltk.org/

[10] [Hockreiter Schimhuber 1997] http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97$_lstm.pdf[Kalchbrenneretal.v32016]h$ $//arxiv.org/pdf/1507.01526.pdf$