Customer Support Chat Bot for Twitter

Omar Walid Hamed Nofal

March 14, 2024

Abstract

1 Introduction

In the recent years, customer support has shifted from phone calls to a more digital service. Nowadays, users expect to find answers to their problems by chatting with a company representative on the internet. Almost all large and medium companies provide a way for users to chat with someone from customer support either on their website, or on social media platforms, like Twitter. Natural Language Processing (NLP) has proven to be well-suited for question answering tasks (QA), as several chat bots have been implemented that are able to take chat history into consideration. This project aims to implement a history-preserving chat bot that can be used by companies to respond to customer inquiries on Twitter.

2 Motivation

Since the rise of the digital age, and the shifting of customer support to websites and social media platforms, companies have been faced with challenges concerning responding to customer inquiries. Mainly, traffic has increased considerably and this puts tremendous load on customer support agents. Also, chatting with a customer doesn't normally happen in one-sitting, instead sometimes it takes hours for customers to respond to agents and vice-versa. This means that agents sometimes have to re-read the whole conversation to remember the context. Also, much of the inquires made by customers can be easily responded to and don't require a human agent to process it. Creating a chat bot which is able to respond to customer inquires would help decrease the load on customer support agents and improve customer satisfaction.

3 Literature Review

Since our problem concerns building a QA chat bot that remembers chat history, we will take a look at the previous research that focuses on either building QA systems, chat bots or both. We will also learn the brief history of neural network architectures in the context of NLP.

Question Answering (QA) systems, are software systems capable of answering questions posed by humans. They can respond to questions concerning facts, definitions, reasons and so on. Early QA systems started off as simple rule-based systems. These systems stored a database of facts or a knowledge base which contained structured information about a specific domain, and as such they were called closed-domain systems. Rule-based systems tried to formulate a database query from the user's input which had to be carefully crafted to make it easier for the system to find information. The information is queried from the database and then presented to the user in a human-readable format. The first rule-based QA systems in the 60s and 70s were BASEBALL and LUNAR. BASEBALL answered questions about Major League Baseball over a period of one year, while LUNAR answered questions about the geological properties of the lunar rocks returned by the Apollo missions. These QA systems were successful but the queries given to them had to be precise so that the model can formulate a database query from the input. By the end of the 90s and the beginning of the 2000s, QA systems started to shift from rule-based to text-based approaches. They leveraged the power of

NLP in order to understand, extract information and formulate answers from unstructured data like web pages, books, and documents. These systems also had the ability to understand unstructured user queries, so that users can be more flexible when asking their questions. One of the earliest most successful systems was IBM Watson which employed advanced NLP algorithms and machine learning techniques to win 1M US Dollars in the popular quiz show Jeopardy!. Since then, most of the effort in developing QA systems almost completely shifted from rule-based systems to machine learning and NLP statistical models. All of this evolution of QA systems happened in parallel or as a side effect of the advancements of chat bots which are now the go-to methods nowadays to implement QA systems. We will briefly go over the evolution of chat bots and how QA systems improved greatly as a side effect.

The first ever chat bot was ELIZA [6] which simulated a psychotherapist. It was very simple by today's standards but engaged users in text-based conversations and demonstrated the potential for machine to interact with humans through natural language. By the 2000s, chat bots started to benefit from the advancements of machine learning and computing power and started to use statistical models and NLP techniques to provide much more natural and realistic conversations than earlier rule-based systems. Recurrent neural networks take credit for much of the advancements in the processing human language and other sequential unstructured information like speech and videos. They are basically like feed forward neural networks but each node maintains a state which it feeds to itself as an input and updates it accordingly. RNNs, however, suffered from the vanishing gradient problem, as the length of the input increases, the model starts to 'forget' the earlier words which was not ideal for long-form text input. This motivated researchers Hochreiter and Schmidhuber to develop the Long Short-Term Memory (LSTM) variant of the RNN [2]. This variant contains much more complex units consisting of a cell, input gate, output gate and a forget gate. These gates regulate the flow of information into and out of the cell, allowing the LSTM to selectively remember or forget information over long sequences.

The latest step in NLP was the invention of Transformers which greatly influenced the development of QA systems and chat bots. Introduced by Vaswani et al. in the seminal paper "Attention is All You Need" in 2017, Transformers have demonstrated remarkable capabilities in capturing long-range dependencies in sequential data, making them particularly well-suited for tasks such as language modeling, translation, and question answering [5].

Unlike traditional recurrent neural networks (RNNs) and LSTMs, Transformers rely solely on self-attention mechanisms to weigh the importance of different input tokens when generating output representations. This attention mechanism allows Transformers to capture global dependencies in the input sequence, enabling more effective modeling of context and reducing the vanishing gradient problem often encountered in RNNs [5].

The effectiveness of Transformers has been demonstrated in various NLP tasks, including question answering, where models like BERT (Bidirectional Encoder Representations from Transformers) and T5 (Text-To-Text Transfer Transformer) have achieved state-of-the-art performance on QA benchmark datasets [1] [3]. In the context of chat bots and customer support, transformers have enabled the development of more sophisticated and context-aware conversational agents. By leveraging large-scale pre-training on vast amounts of text data, transformer-based chat bots can generate more coherent and relevant responses to user queries, enhancing the overall user experience. One of the best examples of a transformer-based chat bot is ChatGPT developed by OpenAI. It is able to mimic real human conversations and answer complex questions from many different domains. QA systems like these are called open-domain systems because they can be queried in a wide range of domains, but they can be fine-tuned to improve their accuracy in a more specialised domain like customer service.

To be able to provide a good user experience, chat bots must be able to remember the history of the conversation which gives the user the feeling that he is talking to a human. This is especially important on Twitter where conversations can span multiple tweets and DMs. Some bots like ChatGPT do that easily by feeding the whole conversation to the user in addition to the user's new query, this way, the bot is able to take the previous history into context. However, other methods exist that allow neural networks to maintain long-term memory. Weston et al. proposed an architecture for a neural network (MemNN) [7] which contains an additional memory component that acts as a dynamic knowledge base. This memory component is updated on every query and queried when generating answers for users. This architecture provided great results on the large QA dataset by Fader et al., however, it suffered a

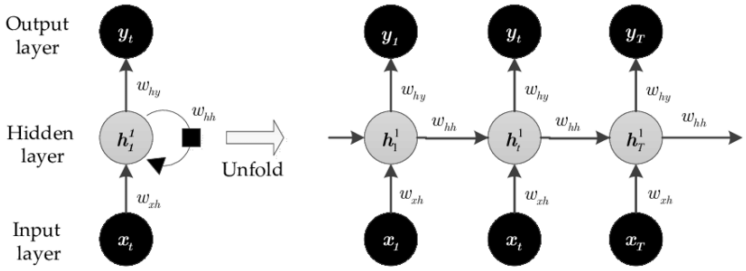

Figure 1: The standard RNN. For each time step t the output of the node at $t-1$ (H_{t-1}) is used as its input.

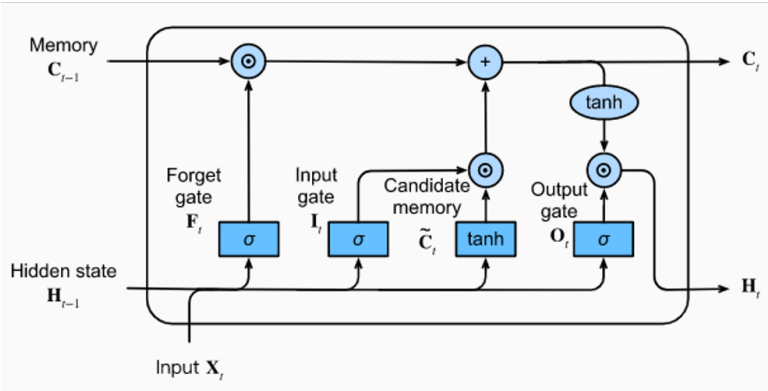

Figure 2: A single LSTM unit. The C is considered as the long term memory component while the H is the same in the case of an RNN. C is modified according to the Forget and Input gate. The H is modified from the Output gate and both C and H are reused at later time steps.

major disadvantage in that it required too much supervision, as the memory layer requires supporting facts to be trained on, but most QA datasets only contain question and answer pairs. Sukhbaatar et al. modified the previous architecture to be trained end-to-end without requiring any supervision on the memory layer, this was possible without sacrificing the performance by much [4].

To conclude, this literature review highlights the significant advancements and challenges in the development of QA systems and chat bots, driven by advancements in NLP and neural network architectures. From the early rule-based systems to modern transformer-based models, the field has seen remarkable progress in enabling machines to understand and respond to human language effectively. Moving forward, continued research and innovation in NLP and machine learning promise to further enhance the capabilities of chat bots, enabling them to provide even more context-aware and personalized interactions in various domains, especially the QA systems domain.

4 Data Analysis

In this section, we will extract some insights from our dataset that will be useful when it comes to training the model. Gaining insight of the data is crucial to be able to understand why the model is working or not. We will focus mainly on statistics useful for the training phase to detect the needed pre-processing that should be applied to our dataset.

The dataset we will be using is the Twitter customer support tweets dataset on Kaggle. It contains 2811774 tweets between customers and famous brands' support agents. Each tweet row contains the sender id, the id of the tweet it is replying to, and the id of a tweet that replies to it (if it exists). In figure 3, we present the top brands by tweet count.

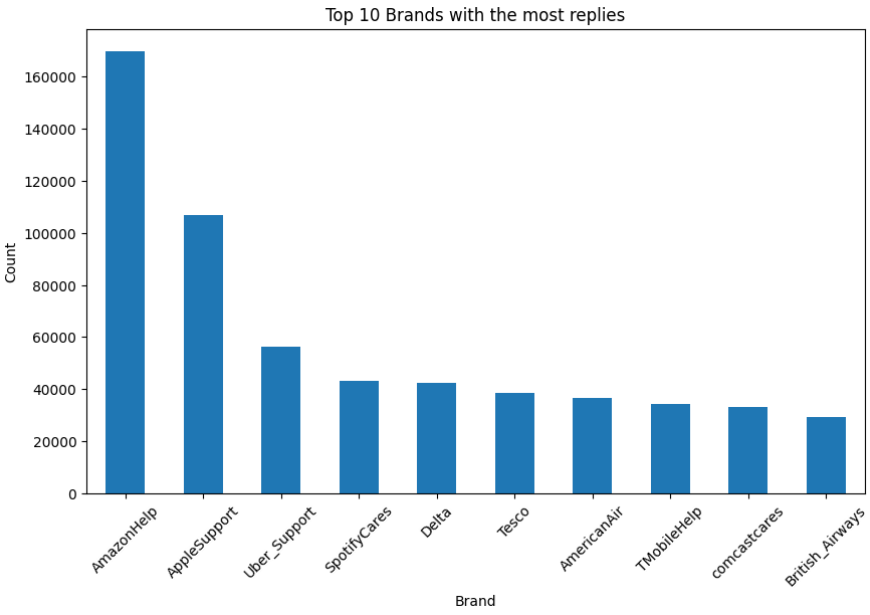

Figure 3: Top brands by tweet replies

4.1 Simple Tweet Analysis

There are 2811774 tweets, the average tweet length is 113 words. The average word length inside a tweet is 5.6 letters excluding stop-words. Almost all tweets are in English except for a few tweets in German and Chinese which will be excluded from training. Overall, the dataset structure is clean and makes it easy to find the replies to a particular tweet which will be useful when generating the training and test datasets.

4.2 Word Frequency Analysis

In figure 4, we can see a bar chart of the top common words after removing stop words. We can see the most famous word is `https`. After investigation, it turns out because a lot of the tweets sent by the companies contained links to other tweets. These links will be definitely removed before the training phase. Other famous words are also `dm` and `us`. Again, a lot of brands would just ask the customer to send them a direct message to assist them in private. These type of tweets would prove difficult to our model and would be unwanted noise and should be removed before training. Figure 5 shows a word cloud of the most common words in our dataset.

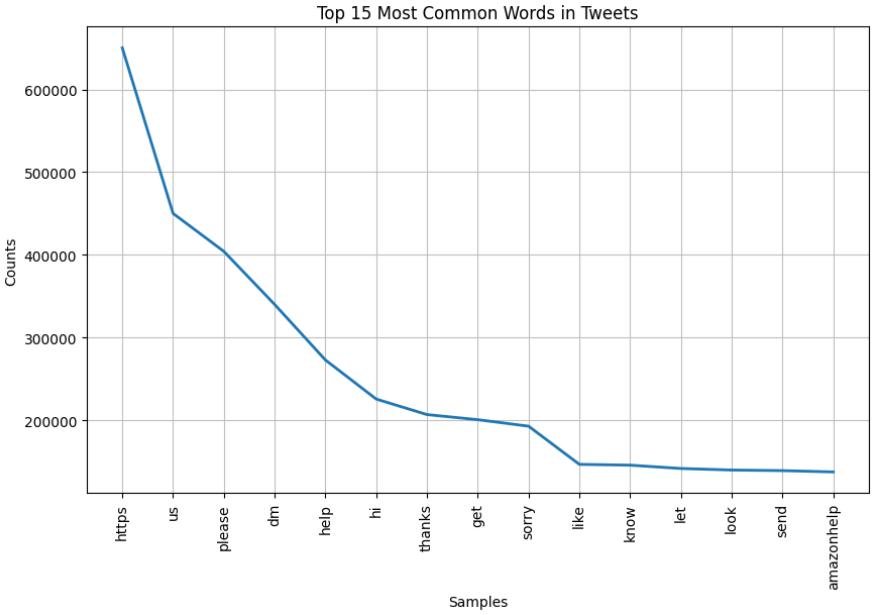

Figure 4: Most common words in the dataset

Figure 5: Word cloud of the most common words

4.3 Bi-gram Analysis

It is sometimes helpful to find the most famous bi-grams in our dataset to detect collocations and to help resolve ambiguities. Figure 6 shows the most famous bi-grams in the dataset. Again the top bi-gram is `(https, co)` which, as explained earlier, is due to the large amounts of twitter URLs sent by brands which will need to be removed. Also, we can see the famous `(dm, us)` bi-gram which occurred in more than 100,000 tweets.

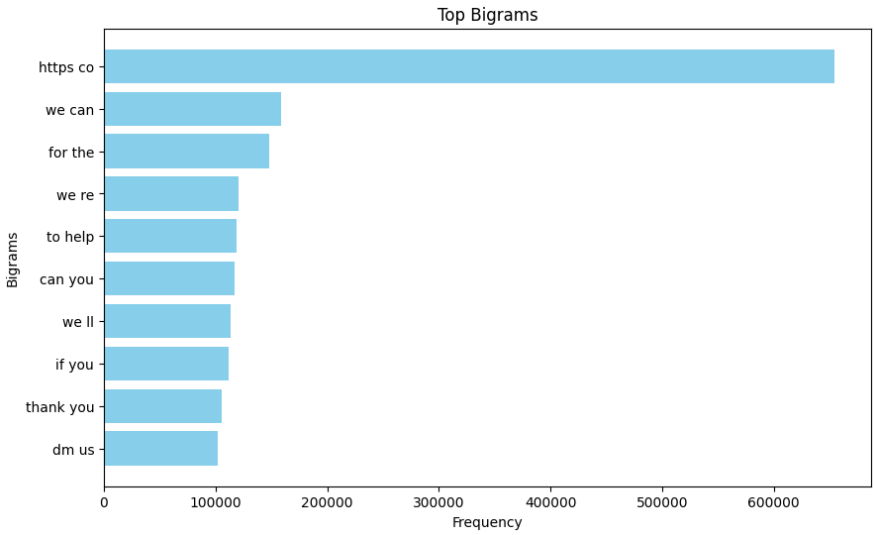

Figure 6: Top bi-grams in the dataset

4.4 Sentiment Analysis

We also applied NLTK's Vader sentiment analysis on the tweets, to try and get an idea of the overall tone of the tweets to see if there are any trends in customer satisfaction but the results were not that interesting. Almost all tweets are neutral with some positive and negative tweets, as evident in figure 7.

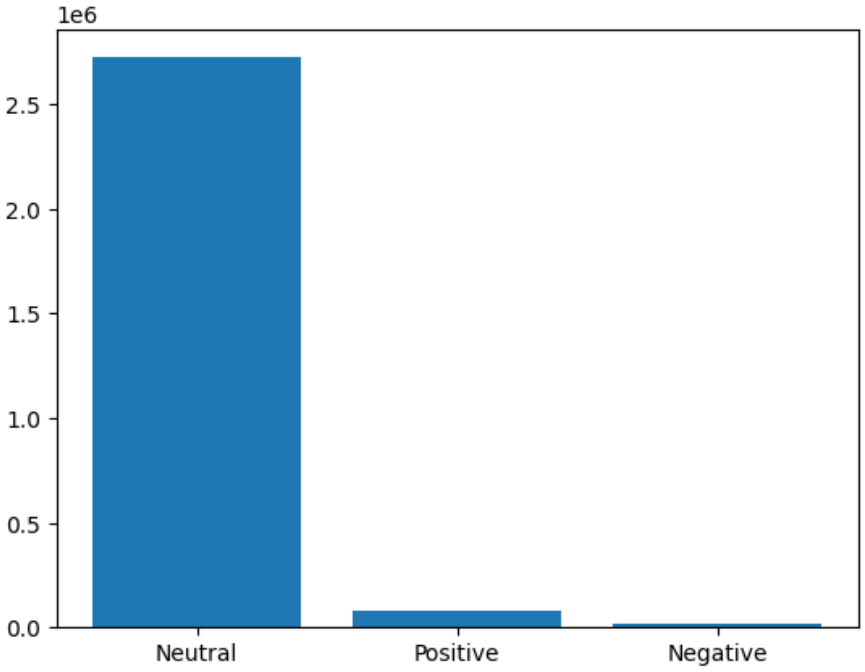

Figure 7: Sentiment Analysis of the tweets

4.5 Limitations

Overall, the dataset is quite dense, organized and clean, however, it suffers from a few limitations.

First, the tweets require a lot of cleaning before being fed into the model for training. In particular, things like other languages, emojis, URLs, id's of the senders, tags, emails, and others, will have to be

removed from the dataset before training.

Second, hundreds of thousands of the responses from the brands are of the type `"Please send us a DM in order to assist you"`. These types of replies are not useful at all and provide a lot of noise for the model, and they will have to be removed, which will decrease the tweet count by a lot, since we will also have to remove the original tweet containing the question asked by the customer.

Finally, the dataset is quite large, and because the model will be created on a personal computer, this might slow down the process considerably. Therefore, depending on the amount of data left after cleaning, some of the data will have to be removed to make the training more computationally feasible

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[4] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks, 2015.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[6] J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, jan 1966.

[7] J. Weston, S. Chopra, and A. Bordes. Memory networks, 2015.