An Experimental Application of Unsupervised Modeling to Chatbots

Christopher Varga and Nicholas Marble
University of Colorado at Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918

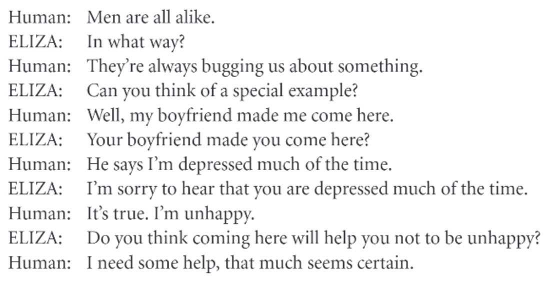
Abstract

Chatbots are computer programs designed to emulate an intelligent conversation that is indistinguishable to what a human could provide. They are used for a multitude of functions from technical assistance to personal companionships. A large variety of these chatbots use large databases with collected hard coded answers or attempt to learn proper answers through some sort of supervised machine learning. These approaches can sometimes create reasonable but unhuman-like responses. We propose to instead use actual human responses derived from Twitter to simulate a more human grammar.

Introduction

Chatbots are used daily around the world for a multitude of tasks. A chatbot is defined by many as a form of artificial intelligence that has the ability to mimic a human-like conversation. This simple idea can easily be implemented to create technical support identities or intelligent personal assistants, and perhaps even for personal companionship. But, to implement a design that can truly be indistinguishable or equivalent to a human is no trivial task.

One of the first known implementations of a chatbot was the program ELIZA. ELIZA was written in MAD-Slip for the IBM 7094. Its name was chosen to emphasize that it may be incrementally improved by its users (Weizenbaum 1966).



Human:	Men are all alike.
ELIZA:	In what way?
Human:	They're always bugging us about something.
ELIZA:	Can you think of a special example?
Human:	Well, my boyfriend made me come here.
ELIZA:	Your boyfriend made you come here?
Human:	He says I'm depressed much of the time.
ELIZA:	I'm sorry to hear that you are depressed much of the time.
Human:	It's true. I'm unhappy.
ELIZA:	Do you think coming here will help you not to be unhappy?
Human:	I need some help, that much seems certain.

Figure 1: An example of a conversation with ELIZA.

ELIZA used keyword mapping, where it searched the input for a word and associated rules with it. Like in the example, the keyword depressed triggers a response of "I'm sorry to hear that you are depressed." While this may work well in certain instances, if the user was to ask a question, this technique would not work as well. To move away from these hard-coded responses you need to implement a type of machine learning environment. Machine learning will allow the program to learn new words or sentences without the need to hard-code these responses. An example of using a machine learning technique can be found in A.L.I.C.E., Artificial Linguistic Internet Computer Entity.

ALICE was created by Dr. Richard S. Wallace in 2003. While learning, ALICE creates an Artificial Intelligence Mark-Up Language which holds in general three different categories generated by a pattern consisting of words, spaces and wild-card symbols (Wallace 2003). These categories are Atomic, Default and Recursive. A template is generated as a response for each category. The Atomic category holds all patterns that do not include the wild card, while the Default category holds all patterns that do have the wildcard symbols. Finally, the Recursive will hold categories that may be too large to accurately deduce a meaning, and hence need to be shortened into smaller ones. For example, Hello * would be categorized in the Default section and a templated response can be generated as Hi there *.

We will be using this type of categorizing with modification that these templated responses will be pulled from the Twitter API to emulate more real human responses. Using this technique will allow for real-time categorization without the restriction of a permanent, static database. These responses will also reflect the currently used phrases and words. The idea is that while a more ridged response or Template may result in a higher acceptance rate from the user, a dynamically generated response will generate a more human like response that makes it less distinguishable as a computer program.

Related Work

Similar work with was done by (Ritter et al. 2010). However, they focused more on the tagging of the Twitter conversations with the approach of unsupervised conversation models, where the discovery of acts amounts to clustering utterances with similar conversational roles. The goal of this was to eliminate the collection and tagging process and allow the learning algorithm to categorize how people converse in a new medium.

Banko and Etzioni instead attempted to use entire websites collected as a corpus for use with ALICE. At the beginning of each learning task, ALICE receives an unexplored concept $c(n)$ as input, instantiates the set of extraction patters (e.g such as) with name of $c(n)$, (e.g. fruit such as), applies them over the input corpus, and uses the assessment model to add high-probability instances of $c(n)$ to its theory (Banko 2007). This is beneficial in the fact that it was almost 80% accurate but is not representative of normal speech behavior.

A reinforcement learning approach to dialogue generation has recently been done by (Li et al. 2016). This approach attempts to take into account the future direction of the dialogue by using rewards evaluated using three factors, namely, informativity, coherence, and ease of answering. The idea is to utilize strategies from improvisation comedy and plays in order to avoid making responses that offer no way of continuing the conversation. In our own approach, however, we mainly seek to achieve more natural responses from the chatbot that approximate what one might find on the Internet by using relevant Twitter responses, and are focused less upon long-term dialogue success. Our approach is discussed below.

Implementation

We begin by receiving a sentence from a user through a command line interface. In our original strategy, we planned to classify the tweets and the input into one of eight broad categories, namely: Status, Question to Followers, Reference Broadcast, Question, Reaction, Comment, Answer, and Response (Ritter et al. 2010). The goal behind this strategy was to have the chatbot produce more accurate responses by responding to Questions with Answers, for example. However, during the implementation we noticed that strictly enforcing this strategy actually caused our chatbot to become less realistic in some scenarios. This is because, in a real conversation, humans often respond to a question with a question. By restricting the chatbot's responses to particular broad categories, we made certain realistic responses impossible from the outset, and the chatbot was therefore too rigid. Therefore, instead, we now focus primarily on finding a tweet in the same narrow category, not what broad category the tweet is in.

The goal of using Twitter tweets as a live corpus is to simulate a more relaxed and realistic conversation, such as what one would find on the Internet. We tried grouping the tweets and the input into similar groups in two different ways. First, we tried grouping the tweets and the user input into a cluster by using the k-means clustering module in the Pattern python library (Smedt and Daelemans 2012). Once the input was placed into a cluster, we then used our RankReponse function to identify and respond with the most similar tweet to the input. The idea behind this approach was to insure that the response tweet was as relevant to the input as possible by being as similar in word order and similarity as possible. However, we found that the results of the basic k-means approach were not effective. The algorithm did not output a relevant response when using a corpus of 100 tweets, and only output slightly more relevant tweets if we increased the number of tweets it could operate on to 1000. However, using 1000 tweets for k-means took upwards of an hour for a

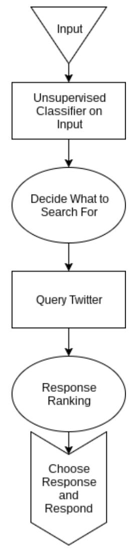

Input
Unsupervised Classifier on Input
Decide What to Search For
Query Twitter
Response Ranking
Choose Response and Respond

Figure 2: Logic flowchart for our chatbot.

single response. This slow time was somewhat mitigated by using a smaller value for k in the algorithm and specifying fewer iterations. However, yielding anything near a relevant response from the chatbot using k-means still took too long to be feasible.

The inaccuracy of the k-means strategy in finding relevant tweets for user input prompted us to take a look at a different strategy, which is now our main approach, namely, Latent Semantic Analysis (Deerwester et al. 1990). This approach takes into account the semantic content of the tweets and input, and treats each tweet as a bag-of-words, where we do not care about the *order* of the words in a tweet, only their *frequency*. All of the tweets and the input are placed into a term-document matrix where each row represents a unique word and each column represents a document (in our case a tweet or the input). Interestingly, since some words in the tweets such as "the" or "a" have a much higher frequency, it could skew the term-document matrix and cause topics to be created such as "the." To mitigate this problem, we eliminated the background noise by assigning a weight to each word using the $tfidf$ formula (or term frequency inverse document frequency). This formula assigns a lower weight to words that are more frequent in other tweets, and a higher weight to words that are less frequent in other tweets:

$$\text{tfidf}(t,d,D) = \text{tf}(t,d)\cdot\text{idf}(t,D)$$

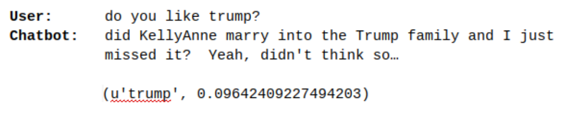
where t is the term in question, d is the document, and D is the set of all documents. The idf formula, in turn is

$$\text{idf}(t,D) = \log\frac{N}{n_t}$$

where $N = |D|$ (the number of documents) and

$n_t = |\{d \in D : t \in d\}|$. Finally, the tf formula is simply the number of how many times a term t appears in a document d.

With the words weighted within the term-document matrix, we then use the Pattern python library to perform Singular Value Decomposition (SVD) on the matrix in order to create a concept space of tweets and terms that are related to similar concepts. This decomposition expresses the term document matrix as a product of three matrices, each with unique properties. In particular, the matrix in the middle is a diagonal matrix where each non-zero value is a concept category. We include the user input into our term-document matrix so that it can also be grouped into a concept category, and then rank the relevance of the tweets in that category using our RankResponse algorithm, as we did for the k-means approach. The tweet that is ranked the most relevant is returned as a response from the chatbot.



```
User:      do you like trump?
Chatbot:   did KellyAnne marry into the Trump family and I just
           missed it?  Yeah, didn't think so…

           (u'trump', 0.09642409227494203)
```

Figure 3: Example of most similar response from 'trump' category

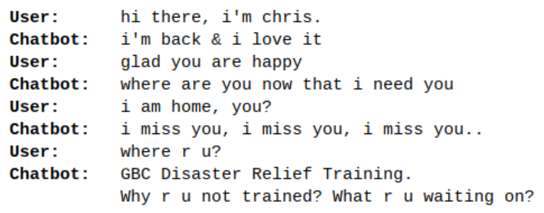
The logic of our RankReponse algorithm is as follows. First, we use the bag-of-words approach and vectorize the user input and tweets according to word frequency. For example, the following tweet A ="It is a nice day is it not" and input B="I think it is a nice day" would be vectorized as follows:

$c =$ [i think it is a nice day not]
$a = [0, 0, 2, 2, 0, 1, 1, 1]$
$b = [1, 1, 1, 1, 1, 1, 1, 0]$

We then calculate the cosine similarity $\frac{ab}{|a||b|}$ between the vectorized input and each tweet. The tweets are ranked according to cosine similarity, and the tweet with the highest similarity is selected as a response. If none of the tweets is similar, that is, if the cosine similarity of all the tweets is zero, then the chatbot returns "I'm sorry, I do not understand."

```
RankResponse($tweets, input$)
1: $R = []$                                ▷ Initialize our list of responses
2: for tweet in tweets do
3:     $r =$ CosineSimilarity(tweet, input)
4:     InsertSorted(r, tweet, R)
5: end for
6: return $R[0]$                           ▷ The most similar result is returned
```

Interestingly, since our corpus is based on Twitter, which is known for slang and typos, the chatbot tends to provide more accurate responses when the user input also contains slang and typos. We attribute this to a higher similarity between the user input and the tweets. An example of an informal conversation utilizing slang from our chatbot is in Figure 4. We were able to expand on this distinction between slang and formal inputs in our evaluation methods.



```
User:      hi there, i'm chris.
Chatbot:   i'm back & i love it
User:      glad you are happy
Chatbot:   where are you now that i need you
User:      i am home, you?
Chatbot:   i miss you, i miss you, i miss you..
User:      where r u?
Chatbot:   GBC Disaster Relief Training.
           Why r u not trained? What r u waiting on?
```

Figure 4: Sample conversation with our chatbot.

Evaluation Methods

We evaluate the effectiveness of our chatbot both qualitatively and quantitatively. Initially, to evaluate our chatbot qualitatively, we asked a specific question, and then determined with our best judgement whether the response from the chatbot seemed human-like. This evaluation method obviously necessarily required a degree of subjectivity, but nonetheless allowed us to judge the relative effectiveness of the chatbot in holding a conversation. In this area, the chatbot performed surprisingly well. Depending on the inputs given, the responses seemed quite relevant. For example, any input including a politician such as Trump receives a remarkably realistic and emotional response such as what one would find on the Internet. Since our chatbot does not keep track of previous questions, the responses sometimes seemed out of context and vague. However, the vagueness of many of the response tweets also helped to keep the conversation open-ended feeling. This was considered a positive by us, since we are aiming for the chatbot to seem less scripted and more natural.

However, this initial strategy for evaluating our chatbot qualitatively was somewhat vague. Therefore, we focused on improving our qualitative evaluations by having five other people judge the relevance of our chatbot's responses to the same inputs. We used two sets of thirty inputs, and had the participants judge each response from the chatbot as relevant or not relevant. The first set of inputs was more formal, and included punctuation and capitalization. The second set was more slang-like, using lowercase letters, "u" instead of "you," and so on. The idea behind this evaluation was to gauge whether the chatbot gave more relevant responses for more formal or less formal kinds of inputs. We predicted that the chatbot would perform better for slang inputs, since it has been our experience that Twitter often contains less formal language. Furthermore, we also tested the ALICE chatbot using these same two sets of questions. The comparison of results between our chatbot and the ALICE chatbot for the qualitative evaluations are located in Figure 5.

Although ALICE was judged to have a higher number of

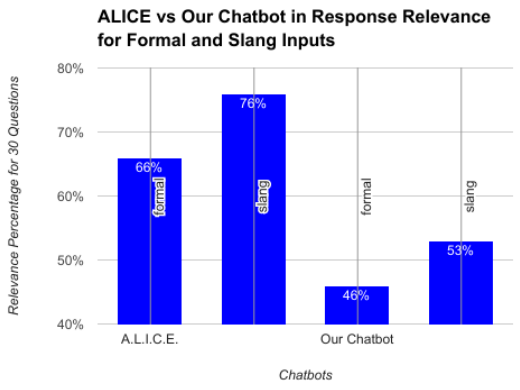
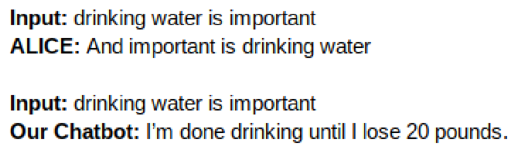


Figure 5: ALICE had a higher number of relevant responses for both formal and slang inputs.

relevant responses according to our participants, we did notice that the responses from ALICE were often less interesting. For example, Figure 6 includes a comparison of the responses from ALICE and our chatbot to the input "drinking water is important."



Input: drinking water is important
ALICE: And important is drinking water

Input: drinking water is important
Our Chatbot: I'm done drinking until I lose 20 pounds.

Figure 6: The ALICE responses often just reversed the order of the input sentence. Our chatbot, however, was often quite humorous.

We attribute the higher relevance rating of ALICE to precisely the fact that ALICE often used the same exact words in a response, so that the response was technically judged to be relevant by our participants. However, we think that even though our chatbot technically had a lower number of relevant responses, it was much more interesting and humorous to interact with. Therefore, we feel that we accomplished our goal of making a more human-like chatbot over other hard-coded answer chatbots such as ALICE.

We also quantitatively evaluate the effectiveness of our chatbot. We originally planned to adopt a method inspired by (Ritter et al. 2010). This method attempts to objectively rate the conversational accuracy of the chatbot in terms of its ability to replicate an actual conversation (in our case, a twitter post and the comments to that post). In other words, we would use a real tweet as input to the chatbot. Then, we expect the response given by the chatbot to be the actual tweets given by humans in a real world conversation. However, we noticed that an easier way to quantitatively measure our chatbot would be to simply measure the number of responses that chatbot made, since it is possible that the chatbot could fail to make a response (when the similarity of all the tweets is zero). We thus quantitatively rate the overall effectiveness of our chatbot by calculating the percentage of responses it makes. We consider our chatbot to be a success if it's response rate is near 80%. The results of our quantitative evaluations before pre-processing are shown in Figure 7.

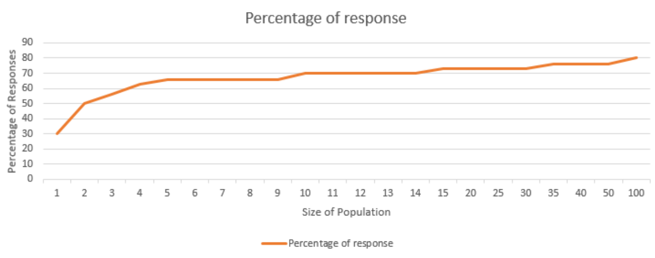


Figure 7: Peak response rate for our chatbot before pre-processing the tweets

We initially used tweets in our database that were raw and unprocessed. That is, these tweets contained symbols such as "@" or "#" and often contained hyperlinks to images or videos. We surmised that we would be able to improve the response rate of our chatbot by pre-processing the tweets to remove these extraneous symbols. We predicted this because of the higher degree of similarity that would likely result between the input and the tweets, since our inputs did not contain symbols. The response rate of our chatbot to formal inputs after pre-processing the tweets are found in Figure 8.

We also made a further general observation about our chatbot's results. As you can see from our two quantitative results graphs, querying a larger number of tweets resulted in a higher response rate overall. We attribute this to the fact that, with a larger corpus to choose from, it is more likely that the chatbot will be able to find a similar tweet within the same concept category as the input. The highest response rate we could achieve with unprocessed tweets was around 73% on average with a population of 100 tweets, and a peak of 80% for 100 tweets. For our pre-processed tweets, we saw a significant increase, with a peak response rate of 86% for 100 tweets, with an average of 76% for 100 tweets.

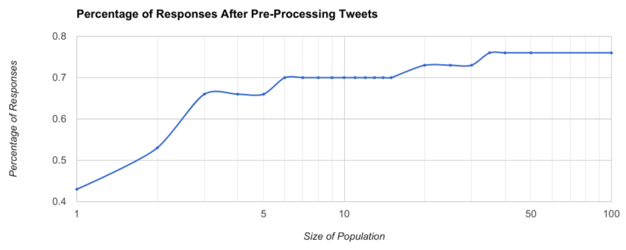


Figure 8: Average response rate for our chatbot after pre-processing the tweets

As you can see from Figure 8, the response rate after pre-processing was higher for lower populations, but still plateaued as the population was increased in a roughly log pattern.

Future Explorations

One problem we noticed with our chatbot was that since we generate responses from our chatbot dynamically for each input by querying Twitter for response tweets, we are not always guaranteed to find similar responses to the input that are relevant. That is because, even with 100 tweets, the probability that there will happen a tweet that is relevant is unlikely. For future work, we therefore had the idea of saving the tweets from our previous queries and storing them in a local database. The idea is to maintain a continuously updating local database of tweets to use for chatbot responses. When the user submits an input, we query for 100 new tweets to add to the local database. We then randomly pick 100 tweets from the database to cluster using latent semantic analysis. By using this strategy, we would ensure that the range of possible chatbot responses is always growing, while still allowing for past responses to be used. We surmise that by using this strategy, we would get an even higher response rate, since the range of tweets we can search through for a response is not limited to only the new 100 tweets we just queried, but also includes older tweets that might have a higher degree of similarity.

A key advantage to our chatbot approach is that it will more closely mimic real human conversations by including slang and other less formal language available in Twitter messages. However, an obvious downside to our approach is that using real verbatim tweets means that the chatbot's responses will be less personalized. For example, currently, our chatbot cannot refer to its conversation partner by a specific name unless, by coincidence, a real tweet happens to contain that name. Therefore, another future area of exploration for our chatbot could include modifying its response so that it is not an exact verbatim tweet. For example, if we know that the chatbot is conversing with someone named Bob, we could inject that name in front of the tweet in specific contexts. In order to accomplish this, we would need to ensure that the name injection resulted in a coherent tweet. This would involve more closely analyzing the parts of speech of the response tweet in order to make accurate injections specific to the context of the conversation and sentence structure, and keeping track of the history of past conversations.

Finally, we think that we could improve the overall human-like element of our chatbot by creating more clever responses than just "I do not understand" when the chatbot cannot find a similar response. For example, we might keep a separate list of conveniently generic responses for when the chatbot cannot find a similar tweet. From that list of generic responses, we might even be able to select the best to respond with by ranking *those* responses by cosine similarity to the input, and if none are similar, responding with a random one. Finally, we could look into having the chatbot simply responding with a completely random tweet from the local database if it cannot find a similar tweet. Or perhaps, if the chatbot cannot find a relevant or similar response from the 100 tweets it queries from the database, to have it select 100 different random tweets to perform latent semantic analysis on. In this way, the chatbot could repeat this process until it finds a relevant response, so that it always responds, instead of merely responding with "I do not understand."

Concluding Thoughts

In this paper, we set out to create a chatbot that avoids using hard coded responses, and in this way simulates a more human-like conversation. To that effect, we think we have succeeded. By comparing our chatbot's responses with those from ALICE, one is able to see that ALICE often merely changes the order of the input in its response. Our chatbot, on the other hand, was often quite human-like due to the humorous and random nature of its responses, and to the fact that it used real tweets from humans. While we do not claim to have created the best chatbot, we do think that our chatbot therefore accomplished at least one of our goals for this paper. [1]

[1] We would like to thank professor Kalita for helpful insights about evaluation method improvements for our chatbot, and other general helpful feedback.

References

[1] Weizenbaum, J. (1966). ELIZA-A computer program for the study of natural language communication between man and machine. Communications of the ACM, vol 9 issue 1 (pp. 36-45).

[2] Wallace, R. (2003). The elements of AIML style. ALICE AI Foundation.

[3] Ritter, A., Cherry, C., & Dolan, B. (2010, June). Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 172-180). Association for Computational Linguistics.

[4] Banko M., Etzioni O., (2007). Strategies for Lifelong Knowledge Extraction from the Web. Turing Center, University of Washington.

[5] Smedt, T. D., & Daelemans, W. (2012). Pattern for python. Journal of Machine Learning Research, 13(Jun), 2063-2067.

[6] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41(6), p.391.

[7] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J. and Jurafsky, D., 2016. Deep reinforcement learning for dialogue generation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1192-1202.