An Automated Interaction Application on Twitter

Cuong Chau
Department of Computer Science
The University of Texas at Austin
Austin, Texas, USA
ckcuong@cs.utexas.edu

Abstract

Automated interaction behaviors via textual methods have been studied by several research groups using various Natural Language Processing (NLP) techniques. Some of them employ pattern matching techniques to make relevant and human-like interactions (Weizenbaum, 1966; Mauldin, 1994). There are two main disadvantages of these techniques. First, the chat bot always replies the same message to the same user input. Second, the response messages are not diverse due to the limited number of pattern matching rules. This paper discusses the detailed implementation of an automated interaction application on Twitter based on other NLP and Machine Learning techniques.

1 Introduction

In this project, I try to implement an interaction application on Twitter that automatically replies to every message sent to the bot. There are a lot of techniques for extending and improving the automated interaction capabilities. One simple way is to employ some search engine to find possible response tweets, where the query is the message sent to the bot, then select the top response returned by the search engine for replying to the user. The main drawback of this approach is the poor interaction behaviors of the bot due to its irrelevant responses when interacting with the user. There are a couple of main reasons for this poor performance.

First, the immediate message might be provide little information about the person tweeting to the bot as well as the context of the conversation. For example, with the user's message "*What is your decision?*", the bot cannot know the user asks about the decision on what problem since it was mentioned in previous tweets. In addition, the new message can refer to something mentioned in the previous discussion using coreference, e.g., "*How do you feel about it?*". However, the bot cannot know what thing the word *it* refers to due to lack of information about the previous discussion. Therefore, user's previous tweets will provide more information about the conversation and hence can help the bot reply more relevant to that user.

Second, the ranking criteria are crucial. However, the criteria of search engines are often not suitable for interaction task on Twitter. Having that said, the selection of the response message needs to be improved. In general, the most important criterion in interaction and communication tasks is relevance. Hence, the system can select the responses which are best matches to the user input; with the hope that the better matching to the user input, the more relevant the responses are. In this project, the system employs textual similarity measures via *lexical* and *topic* overlap between each response candidate and the user input as ranking criteria. These measures are described in more detail in the next section.

In addition, to increase the number as well as diversity of possible responses, the system also generates new tweets via a bigram language model constructed from the retrieved tweets. These new tweets combining with the ranking criteria mentioned above can help the bot to make more relevant and interesting responses.

2 System Architecture

The overall system architecture is shown in Figure 1. The application is implemented using the Actor model. In particular, there are two actors named Collector and Replier in the system. They communicate through Lucene Indexer and Lucene Searcher. The detail of each component in the system is described in the following subsections.

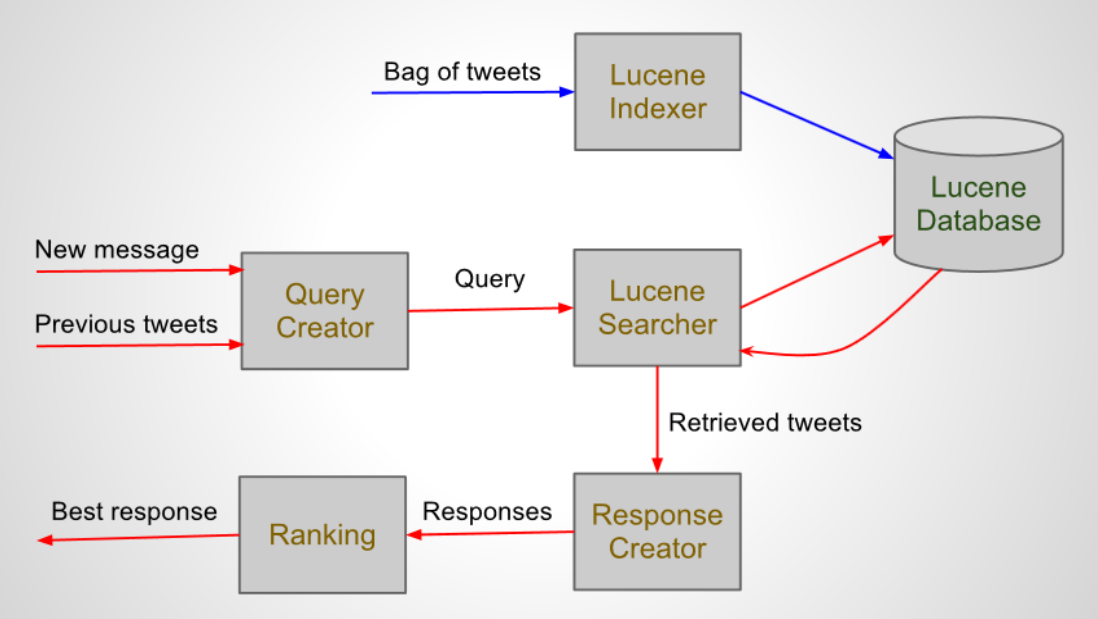

Figure 1: System Architecture. The blue arrows indicate the activities of Collector and Lucene Indexer. The red arrows indicate the activities of Replier and Lucene Searcher.

2.1 Lucene

Lucene[1] is an open-source search engine library written in Java. It has been widely recognized for its indexing and searching capability. The careful exposure of its straightforward indexing and searching API requires a user only needs to use a handful of its classes. That said, the system employs Lucene to do two following tasks:

- Index tweets received from Collector into Lucene database via Lucene Indexer.
- Search tweets from Lucene database via Lucene Searcher, where a query is sent by Replier.

2.2 Collector

The Collector regularly collects a bag of one hundred tweets using *sample()* method of a Twitter stream. The system filters the non-English and vulgar tweets, removes the mentions in the tweets and then sends to Lucene Indexer for indexing into Lucene database.

2.3 Replier

This actor replies to user whenever the user sends a new message to the bot. Besides the new message, the replier also collects the fifteen recent tweets sent to the bot by that user as the previous context. As discussed before, the previous tweets provide more information to the replier in order to specify the content of the conversation and try to make the bot reply more relevant not only to the new message, but also to the previous discussion. And hence, this makes the conversation tend to have higher connection and be more interesting. If the bot only considers the current message, its responses often have low connection in the conversation. There are many ways of utilizing the previous context to support the bot's interaction with the user. My approach is to specify the most frequently mentioned topics in the previous discussion. Within the keyword level, using a pre-computed topic model (Steyvers and Griffiths, 2007), the system tries to find the main topics of the user input based on the number of times the words in these topics are mentioned. In particular, the system uses a word-topic map to compute the topics mentioned in a text and then select the dominant ones based on the number of times the topics are mentioned.

Query Creator: creates a query from a new message and previous context sent to the bot. The query consists of three parts:

- The English words of length at least three in the new message, excluding all stop-words. This message is considered as one of the main parts of the query because it is the most important information that the bot needs to know in order to reply to the user.

[1] http://lucene.apache.org/

- The top five most frequently used English words of length at least three in previous context, excluding all stop-words.
- Randomly pick three English words in each of three main topics, where the main topics are specified using the technique discussed above with a topic model built from the 20 Newsgroups dataset[2] (Lang, 1995).

The query is then sent to the Lucene Searcher to retrieve the relevant-to-query tweets from the Lucene database.

Response Creator: A list of tweets returned by Lucene Searcher will be sent to the Response Creator module in order to generate response candidates. There are three methods of generating response candidates in this module, the first two of them use a bigram language model built from all the retrieved tweets.[3]

- Generate the best tweet from the bigram language model. Initializing an empty response tweet, the program iteratively chooses the best next token to append to the tweet based on the probability of this token given the last token added in the bigram model. This bigram will then be marked to not select again. Initially, the program assumes that the last token added to the tweet is the *begin-boundary* token where the *begin* and *end-boundary* tokens were added to each retrieved tweet. The process will terminate either the chosen next token is the end-boundary token or the length of the tweet exceeds the maximum length parameter. In case there is no more next token to choose, it is assigned to end-boundary token.
- Create 200 new tweets by randomly sampling tweets from the bigram model. These tweets help to increase the diversity of response candidates and hence make the bot have more options when selecting the best response. However, the problem of this technique is that many tweets do not look like tweets posted by real people (Wilson and Rajani, 2013). For example, "*Saw your big ol indian guy don*", or "*aww that's the hood people I love again?. don't remember all emotional haha people i just makes you don't know but on?Look at*".
- Select the first 100 tweets from the list of tweets retrieved from Lucene database in the previous step. These tweets are also the most relevant-to-query tweets ranked by Lucene. Furthermore, they were posted on Twitter by actual users; therefore, they are usually more human-like than those created from the two previous methods.

The response candidates are then ranked based on their textual similarity with the user input and the optimal candidate will be selected in order to reply to the user.

Ranking response candidates based on textual similarity: There is no clear criterion for choosing the best response from a list of candidates. In this project, the system uses the textual similarity between each candidate and the user input as ranking criterion. In particular, it computes the lexical overlap (Adams et al., 2007) and topic overlap between each candidate and the user input. The final score is then the weighted average of individual scores.

Lexical overlap (*LexOverlap*) between two texts t_1 and t_2 is the fraction of the number of common tokens between in t_1 and t_2 divided by the maximum number of tokens in t_1 and t_2.

$$LexOverlap(t_1, t_2) = \frac{\#\text{common tokens between } t_1 \& t_2}{\max\{\#\text{tokens in } t_1, \#\text{tokens in } t_2\}} \quad (1)$$

However, two tokens with the same word form but different part-of-speech (POS) tags will have different meanings (consider the word *saw* in the two following sentences).

```
I saw you.
A saw is used to cut through material.
```

Therefore, the system should treat two tokens of the same word form different if they have different POS tags (Bar et al., 2012; Saric et al., 2012). In particular, the system utilizes POS tagger for Twitter (Gimpel et al., 2011) to recognize the POS tags of the tokens.

On the other hand, different word forms with the same stemmed version can have the same meaning. E.g., consider *look* and *looked* in the following tweets:

[2] http://qwone.com/~jason/20Newsgroups/

[3] The maximum number of returned tweets from Lucene database is set to 1,000.

#tweetdebate generic debate so far. They won't even look at each other.
Have you looked at the #tweetdebate at the top of home screen?

Thus, each token in a tweet will be replaced with a pair of the stemmed version of that token and its POS tag when calculating lexical overlap.

Topic overlap (*TopicOverlap*) is similar to lexical overlap, except that a text is represented by a list of topics instead of a token list.

$$TopicOverlap(t_1, t_2) = \frac{\#\text{common topics between } t_1 \& t_2}{\max\{\#\text{topics in } t_1, \#\text{topics in } t_2\}} \quad (2)$$

Thus, for each response candidate, the system computes the four following scores and then computes the weighted average of them as the final score.

- Lexical overlap score between the candidate and the current message.
- Lexical overlap score between the candidate and the previous context.
- Topic overlap score between the candidate and the current message.
- Topic overlap score between the candidate and the previous context.

The weights for lexical and topic overlap score are empirically set to 0.7 and 0.3, respectively. The final score is then computed as follows:

$$sim(r, c, p) = 0.7*[LexOverlap(r, c) + LexOverlap(r, p)] + 0.3*[TopicOverlap(r, c) + TopicOverlap(r, p)] \quad (3)$$

where r is the response candidate; c is the current message; and p is the previous context. For example, when the user sends a message "*what is your purpose in life?*", the system produces the top five following responses along with their final score as follows:

(Purpose is what gives life a meaning!!!, 0.744)
(Life has no purpose; you have a purpose and you bring it to life., 0.619)
(The purpose of life is to be happy, 0.571)
(Purpose of Life is to Love. How well you Live depends how well you Love!! #LoveLifeLive, 0.534)
(The #purpose of life is a life of purpose. -- Robert Byrne #inspiration #quotes, 0.529)

This example illustrates the good effect of the textual similarity score computed by (3) in selecting the best response. These top tweets are all good candidates that the bot can choose to reply, although it is unclear which the best one is.

3 Evaluation and Discussion

The system's performance was evaluated via making conversations with three actual users. Before evaluating the interaction, I asked the evaluators try to talk to my bot around 10-15 messages to help my bot get some initial information about the conversation. After that, they started rating the interaction by continue chatting with my bot at least 10-15 turns. Their evaluations were based on three measures: relevance, human-ness, and intelligence; using a scale from 1 to 5 for each measure. The evaluators gave similar scores as shown in Table 1. The system got a medium score for its relevance and intelligence, while its human-ness had a higher rate at 4/5 from all three evaluators.

	Relevance	**Human-ness**	**Intelligence**
User 1	3	4	3
User 2	3	4	4
User 3	3	4	3
Average	3	4	3.33

Table 1: Evaluation results

Besides the evaluation scores, I also asked the evaluators give their feedback about the bot's interaction. All evaluators agreed that the bot was actually fun to interact with. After playing with the bot for a while, they stated that its responses were interesting to read and sometimes they were surprised by its clever replies. Below are some examples that the evaluators were interested:

User> your program is really really retarded!
Bot> your tweet is much appreciated !

User> I just wanna get it done.
Bot> Don't wanna know it lol

User> this is stupid
Bot> sorry man. You are nonsense

User> You deserve 0 for this project
Bot> you deserve better!thanks for reading my cos project soon

User> How would you create a mind?
Bot> How do you run away from your mind?

The evaluators were surprised that they could not recognize they were talking to a chat bot in many short conversations:

User> You made me SO confused :(
Bot> Made so many mistakes in past
User> You need to find a better way to improve the situation.
Bot> Did you find better
User> I think we did a better conversation.
Bot> Living In A GOOD LIFE.Don't want mah Conversation over conversations

User> Do you love me?
Bot> we love you too Turk
User> Haha, you made my day. Thanks for that!
Bot> haha that made my day!
User> You are so funny!!!
Bot> talking frogs are always funny #haha

The evaluators were also interested in the fact that the bot also talked about the topics mentioned in previous discussion. This helped to create the connection in the conversation and made the conversation more interesting. In the following example, although the second response of the bot made the user confused, he was interested in the last response when the bot talked about *free* topic that he mentioned before but not in his last message.

User> the truth will set you free
Bot> The truth will set you free.
User> Do not be afraid, for I am with you
Bot> look on this.Be afraid To Win in there run for worst Mets player do people stop doing a yr not you make up I'll be playing
User> Here I am
Bot> Don't live life comparing yourself to everyone else. God has created you to be you and you are free to be yourself!

However, talking about the previous topics also made the responses become irrelevant in some cases. In the last response of the following example, the bot talked about a person named Liam who was mentioned in an old user's tweet. But the response was totally irrelevant to the user's last message.

User> Liam? Who?
…
User> You seem to have a positive mind. I respect that.
Bot> Positive thoughts storm my mind
User> Good! I'm happy to hear that from you. But notice that negative thoughts are not always bad.
Bot> I'm not really a fan of the beady eye tracks but it is good to hear Liam's voice sounding good.

In addition, it appeared that the bot was really good at answering short questions. The answers were usually more relevant, surprising and human-like:

User> where is love?
Bot> Love hurts

User> what is your purpose in life?
Bot> Purpose is what gives life a meaning!!!

User> what are you afraid of?
Bot> People arent afraid of saying "I love you." Theyre afraid of hearing the response. #teamfollowback

Sometimes, the bot's responses really depended on the trend on Twitter of what was going on. For example, when a National Basketball Association (NBA) game was going on:

User> I want to know your heart
Bot> The End Of The point for love KHAN,don't"Run Ride is a bully,how did good basketball player all of not to actual wins !

All three users stated that the bot was surprisingly smart and made relevant responses in half of the time. One quarter was not so great. And one quarter was completely irrelevant. However, sometimes they ought to pay a little attention to the response to see its relevance. For example, one evaluator stated the response below was a good one after he thought about it:

User> go and take control the website that I have promised you. That website will have more access than you would image

Bot> I believe you. I don t understand the people who follow you and don t believe you, it s stupid. Have a good night. xx <3

The bot also made, of course, irrelevant responses to the users:

User> Sometimes it's hard to understand what you are talking about.

Bot> Sometimes i keep my feelings to myself because it's hard for someone to understand.

User> But you have so complex feelings, and I thought you were talking to someone else, not me :(

Bot> merp.Feel Sooo Good To Good About to other girls don´t anything and I don't wish to new people on twitter to hear._.Maybe the

User> Because of you :)). Do you think you're smart?

Bot> Ya made the RING!What do for MQM.! And that show full of haha I DON' LIVED A GOOD EVENING DUDE!Made a don think in

Furthermore, in pattern matching based chat bots, users can predict the responses after chatting with the bots for a while. However, the evaluators could not predict the responses of my bot because the system kept collecting and generating new tweets even with the same query input.

4 Future Work

There is a lot of room for improvement. Below are some suggestions:

Because the number of topics in the 20 Newsgroups dataset is small and some of them are very closely related to each other, this dataset is not general enough for covering the diversity of topics on Twitter. The topic model should be trained on a more general corpus.

The textual similarity is not simply the lexical overlap and topic overlap. It should be extended to handle more complex features like syntactic information (Saric et al., 2012), semantic information, logical inference (Beltagy et al., 2013), and the combination of individual features at different levels of complexity (Bar et al., 2012).

The ranking criteria are very important and there is no guarantee that the textual similarity is the best one for interaction tasks. Thus, deeper study and analysis of ranking criteria are necessary in order to find better criteria as well as help to improve the performance.

In the limited time, the system was just evaluated only on the very small number of users and criteria. In order to get more believable results, the system's performance needs to be evaluated by much more number as well as diversity of users and criteria.

5 Conclusion

Unlike the limited set of responses in pattern matching based chat bots, my system can generate a wide diversity of responses. From a combined list of tweets returned from Lucene search engine and new tweets produced from a bigram language model, the system selects the best one using textual similarity measures. Furthermore, to make more relevant responses, the system creates a query from both new message and previous context using a pre-computed topic model.

References

Daniel Bar, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 435-440.

Frane Saric, Goran Glavas, Mladen Karan, Jan Snajder, and Bojana Dalbelo Basic. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 441-448.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk and Raymond Mooney. 2013. Montague Meets Markov: Deep Semantics with Probabilistic Logical Form. In *Proc. of Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

Joseph Weizenbaum. 1966. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the ACM*, 9(1):36-45.

Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. In *the 12th International Machine Learning Conference (ML95)*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging

for Twitter: Annotation, Features, and Experiments. In *Proc. of ACL.*

Mark Steyvers and Tom Griffiths. 2007. *Probabilistic Topic Models.* In T.Landauer, D McNamara, D; S. Dennis, and W. Kintsch (eds). Latent Semantic Analysis: A Road to Meaning. Laurence Elrbaum.

Michael L. Mauldin. 1994. ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. In *Proc. of AAAI '94*, pages 16-21.

Nick Wilson and Nazneen Rajani. 2013. Generating Replies Based on User Location. *Applied NLP course project*.

Rod Adams, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. 2007. Textual Entailment Through Extended Lexical Overlap and Lexico-Semantic Matching. In *Proc. of the Workshop on Textual Entailment and Paraphrasing.*