A Personal Conversational Model

Lawrence Lin Murata

Ana-Maria Istrate

Computer Science Department, Stanford University

lmurata@stanford.edu

aistrate@stanford.edu

I. Introduction

Conversation with machines has been an important topic in both research and fiction for a long time. The possibility of having conversations with machines has always captured the imagination of people. In 1950, Alan Turing proposed the famous Turing test to answer the question: "Are there imaginable digital computers which would do well in the imitation game?". Since then, the Turing test has become a highly influential concept in the philosophy of artificial intelligence.

The idea of talking with machines has become popular with movies like *2001: Space Odyssey* and *Her*, and with product like Apple's Siri, Microsoft's Cortana, Facebook's M and Google Now. It makes sense for natural language to become the primary way in which we interact with devices in the future, because that's how humans communicate with each other. Thus, the possibility of having conversations with machines would make our interaction much more smooth, natural and human-like. In this project, we explored different methods that model distinct aspects of language.

In our conversational model, we take the human utterance as an input (for instance, "How are you?"). We then use a Support Vector Machine (SVM) with a linear kernel to generate the machine response word-by-word, where we start off by predicting the first word in the response, and then each predicted word, together with the human input, is used to predict the next word until we generate the entire response. The final output is the machine-generated response (for instance, "I am good").

The main method we develop, the Recurrent Statistical Model (RSM), relies heavily on machine learning knowledge and is being used for CS 229 (Machine Learning). Our other methods, Logic Model and N-gram Search, while also relevant for a machine learning class, are more focused on CS 221(Artificial Intelligence) material.

II. Related Word

A. Chat Bots and Conversational Agents

Over the last decades, researches have worked on creating chat bots and conversational systems, but these complex systems often require a processing pipeline of multiple stages (Lester et al., 2004; Will, 2007; Jurafsky & Martin, 2009). Besides requiring a complex processing pipeline, many of the classical approaches for modelling conversations rely on language rules or domain-specific knowledge, usually provided by the creator of the system. Our goal was to create a conversational model that could work reliably without requiring huge data sets or any previous knowledge about the domain, the language, or any other sort of pre-computed knowledge.

B. Neural Networks for Natural Language Processing

The more recent success with statistical approaches have inspired us to develop our own probabilistic model. The works of Bengio et al. (Bengio et al., 2003), Mikolov et al. (Mikolov et al., 2010) and Mikolov (Mikolov, 2012) have successfully demonstrated the application of recurrent neural networks to model languages. Long Short Term Memory (LSTM) recurrent neural networks have become prominent choices for learning language models. In 2015, Sordoni et al. (Sordoni et al., 2015) and Shang et al. (Shang et al., 2015) were able to model short conversations using recurrent neural networks that were trained on short chats.

C. A Purely Statistical Approach with Recurrent Neural Networks

More recently, Google researchers O. Vinyals, Q.V. Le (O. Vinyals, Q.V. Le, 2015) developed a conversational model that makes use of the sequence to sequence (seq2seq) model described in (Sutskever et al., 2014), using recurrent neural networks to read the human input, token by token, and predict the machine-generated response, token by token. It's an interesting paper because it proposes a system that is simple than most of the models proposed before. The neural conversational model is a purely statistical model that is general and doesn't require any domain knowledge. The model, however, relies on rather large data sets — an IT Helpdesk Troubleshooting data set with 30M tokens and an OpenSubtitles data set (Tiedemann, 2009) with 62M sentences (923M tokens) for training and 26M sentences (395M tokens) for validation. Since the implementations we were able to find online did not perform as well as expected with smaller data sets, we decided to create a simpler statistical model that could generate conversations with considerably less data (as you will see, our data sets have 672 to 19,740 lines).

A drawback that all statistical approaches naturally have is that they oversimplify many aspects of communication in order to frame a conversation as simply finding the most likely sequence as a response. For example, it does not capture general knowledge about the world, human logic (which our Logic Model captures), the essence of human communication (with its goal to exchange information), or a consistent personality.

III. Data and Feature Engineering

A. Datasets

We used three main data sets, all of them containing conversations from Friends, the TV series. The first smaller data set (data set A) contains a small portion of the script from the second season of Friends. The raw data set contains 672 lines. The second data set (data set B) is composed of the first and second seasons together. The raw data set B contains 12,380 lines. The third data set (data set C) has seasons 1, 2 and 3 combined. The raw version of data set C has a total of 19,740 lines. We used the three data sets and compared the results for each of them in order to understand how our model improved as we increased the size of the data sets.

For testing, we used a set of 42 questions in 4 different domains: general conversation (21 questions), philosophical questions (5 questions), feelings (11 questions) and closing (5 questions). It's important to remember that our data set was very general and not specific to any of these domains, because our goal was to develop a general-purpose conversational model that can perform well under different domains and situations.

B. Processing

We parsed the data sets in order to remove character names that were present in the beginning of each line, punctuation marks, remove introductions or section titles that were not part of the dialogues, and descriptions of the scenario or of visual aspects of the scene such as body movement. We also chose to lower case all of the text.

CLOSING CREDITS

[Scene: Chandler and Joey's, Joey is reading a script as Ross enters]

ROSS: All right I've been feeling incredibly guilty about this, because I wanna be a good friend, and dammit I am a good friend. So just, just shut up and close your eyes (kisses Joey).

JANICE: OH.....MY.....GAWD!! (Chandler rushes over and kisses her)

Fig. 1 Example of our raw data. It contains punctuation marks, upper case characters, and descriptions of the scene

all right i've been feeling incredibly guilty about this because I wanna be a good friend and dammit i am a good friend so just just shut up and close your eyes

oh my gawd

Fig. 2 Example of the processed data resulting from the raw data in fig. 1

Following a bag-of-words approach, we then used a combination of uni-grams and bi-grams, together with TF*IDF, as the inputs in our Support Vector Machine. Using TF*IDF helped us understand how important each word is in the documents. We experimented with different n-grams to capture important long-range correlations between the words.

IV. METHODS

A. Logic Model

At an abstract level, one fundamental thing a good personal assistant should be able to do is to take in information from people and be able to answer questions that require drawing inferences from the facts. In some sense, telling the system information is like machine learning, but it feels like a very different form of learning than seeing 10M images and their labels or 10M sentences and their translations. The type of information we get here is both more heterogenous, more abstract, and the expectation is that we process it more deeply (we don't want to have to tell our personal assistant 100 times that we prefer morning meetings).

Our logic model parses human utterances into first order logic rules. By enforcing logic rules, the model is able to draw inferences from facts given by the human. Then the model generates responses and answers questions according to the inferences it can draw from facts it has previously learned. Our logic method models language according to logic. It is very limited compared to statistical approaches, yet efficient for some and fast. Another advantage of the model is that it doesn't require any data or training since it relies solely on logic to generate responses. *Disclaimer: the starter code for this model comes from CS221, an Artificial Intelligence class, but it was extended for this project.*

B. N-gram Search

We defined our model as a search problem, where we are searching for the words that will be part of our reply. Thus, in our framework, a state is defined as an ordered set of words that could be added to the reply. Each state has one more word than the previous one. We have started tackling the task of generating replies by using a very basic n-grams model. We start by choosing a word that starts the reply and a number of words we want our reply to have. The starting word is our initial state. Right now, we are choosing the starting word ourselves, but in the future, we plan on generating the starting word based on the last message (the one that we are replying to). We are also manually inputting the numbers of words we want in our reply ourselves, but in the future we want to account for that by using grammar and stopping the reply when a punctuation mark is encountered.

Given a starting n-gram, this model finds the next n-gram that is most likely to follow the previous n-gram in the sequence until the response reaches a certain predetermined length. For example, to start with, we find all n-tuples (in our case, 2 or 3) in our dataset that begin with the starting word. Then, we choose the n-tuple that happens with highest probability. Then, we do the same thing, but match the tuples that start with the previous n-1 words and take the one that occurs with highest probability. We keep repeating the process until we generate the number of words that we want. To know when to stop, we are keeping a level that tells us how many words we have added to the reply. We are stopping when the level equals the desired number of words in the sentence.

C. Recurrent Statistical Model

Our statistical conversation model is the one that allowed us to generate the most flexible conversational models, even though responses naturally didn't always follow a correct grammatical structure. We developed the RSM model, which uses the words in the human utterance as inputs in a bag of words (TF*IDF and a mix of uni-grams and bi-grams). With the bag of words, a Support Vector Machine (SVM) with a linear kernel predicts the next word in the machine's response. It then adds the predicted word into the bag of words in order to predict the next word. The process is repeated until the <EOS> tag is predicted, indicating the end of the machine's response.

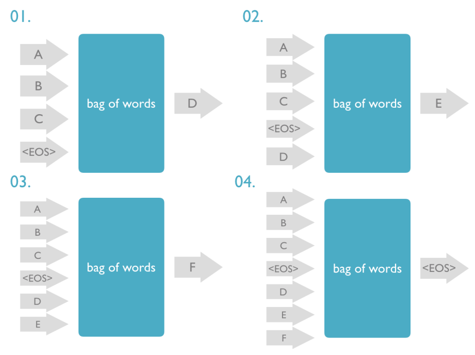
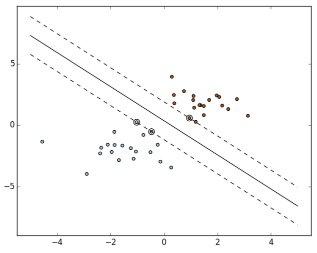


Fig. 3 RSM is a purely statistical and recurrent model for building conversations. In this example, RSM receives a human input "A B C" (where <EOS> indicates the end of the utterance) and, token by token, predicts the machine response "D E F" in four steps

In the example portrayed in figure 3, the human input is "A B C" and the machine-generated reply is "D E F." The <EOS> token indicates the end of an utterance. The bag of words is actually a mix of uni-grams, bi-grams and TF*IDF used as inputs for our Support Vector Machine. This algorithm constructs a hyper-plane or a set of hyper-planes in a high or infinite dimensional space. In figure 4, the three points under the two dotted lines are called support vectors. The distance to the nearest training data points is called the functional margin and, in general, the larger the margin, the lower the generalization error. Thus, intuitively, a good classification (with higher confidence scores) is achieved by a hyper-plane that has the largest functional margin.

Fig. 4 A visualization of how Support Vector Machines work



Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, in two classes, and a vector $y \in \{1,-1\}^n$, our Support Vector Machine solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

subject to $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i$,

$\zeta_i \geq 0, i = 1,...,n$

Its dual is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

subject to $y^T \alpha = 0$

$0 \leq \alpha_i \leq C, i = 1,...,n$

Where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$. Where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ. The relation between C, alpha and the number of samples is given by $C = \dfrac{n\_samples}{alpha}$.

The decision function is:

$$\operatorname{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

V. Experiments, Discussions and Results

A. *Parameters and Techniques*

There are three major models that we tried in approaching our problem: n-grams search, a logic model and multi-class classification. We focused on the multi-class classification for this project.

1. Multi-class classification approach

Our model fit a multi class classification problem approach. At each step, the classified variable was a sequences of words, the "bag of words" that we initially generated after each reply and kept updating by appending the previous generated word. The labels were words that the bag of words could map to. Thus, for each reply we would have several possibilities of choosing the next word from the set of possible words. Since multi-class classification is a model that works for classification problems with more than two classes, this approach fit our problem the best. In order to get the best results, we have tried different classifiers with datasets of different sizes. We have obtained the best results by using a Support Vector Machine (SVM) kernel and a dataset consisting of the first three seasons of the "Friends" TV Show. Below we talk about the two important steps that we took: training and testing and how the results varied with each of the factors:

We tried three different classifiers: Support Vector Machine (SVM), Multinomial Naive Bayes and Logistic Regression. Below, we give an overview of each of them and explain the benefits and limitations.

i. Support Vector Machine(SVM) with a linear kernel

The model that gave the best results was the Linear Support Vector Classifier, which is a Support Vector Classifier that has a linear kernel. The reason why we chose a Support Vector Machine classifier with a linear kernel is because it scales better to large number of samples. Indeed, we saw better results with increasing the size of dataset and the number of text samples. This is because Linear SVC has more flexibility regarding the penalties and loss functions it uses than the other classifiers. We tried this classifier under by setting the multi class strategy parameter to a One-vs-Rest, a model that fits one classifier per class. Basically, for each classifier, the class is fitted against all other possible classes. A benefit of One-vs-Rest classifier is its efficiency, being able to generate predictions in considerably less time than all the other classifiers we tried. We run the algorithm for a maximum number of 1000 iterations, and we have used an L2 regularization penalty that prevents overfitting and a squared hinge loss that optimizes the loss function.

ii. Multinomial Naive Bayes

Naive Bayes is a supervised learning algorithm that assumes independence between pairs of features. Multinomial Naive Bayes is an algorithm usually used for multinomially distributed data such as in text classification. For each class y, the distribution is parametrized by a vector $\theta y = (\theta y1 + \theta y2 + ... + \theta yn)$, where n is the number of features that are being used. In our case, the features being used are the sequences of words (the "bag-of-words") that we are trying to predict, so it would roughly be equal to m^2 where m is the number of replies of the dataset.

$\theta yi = \frac{Nyi + \alpha}{N + \alpha n}$ where Ni = $\sum_{x} xi$ (the number of times feature i appears in class y)

N = $\sum_{x} Ni$ (the sum of all features for class y)

α = smoothing parameter that accounts for non-present features

This learning algorithm worked relatively well on small datasets, and considerably worse on larger ones. The major problem with applying Naive Bayes to our model is that it assumes of independence between words, which is not a valid assumption to make on larger datasets. In this case,the prediction would just be composed of a set of words that happen individually with the highest probability, not accounting for the context in which the words appear, which becomes very important especially with increasing the size of the dataset. However, when working on smaller datasets, assuming independence between words is a semi-valid assumption to make, because there are fewer choices to make when choosing a class, so the probability of predicting the choice that makes sense in the context is higher.

iii. Linear Regression

We have also tried multinomial regularized logistic regression under the One-vs-Rest approach. The model did not give very good results, always predicting <EOS>, an element that we chose to denote the end of sentence.

Metrics

Quantitative

Granular Conversation Test (GCT)

The traditional, well-renowned qualitative metric for testing a chatbot is the Turing test, which looks at its level of "humanness". Since this is a very binary test that does not account for other qualities of the model apart from "seeming human", we created our own qualitative metrics, in order to assess better a conversational instance in several areas that we found relevant, apart from "humanness". We have created and used the Granular Conversation Test (GCT), that tests a reply given back by the computer in 4 areas:

1. Semantics - Does this reply make sense by itself?
2. Structure - Does this sentence have structure?
3. Context - Does this reply make sense in this context?
4. Humanness - Does this reply seem like it's coming from a human?

Each reply is given a score of 0 or 1 in each of the four categories. The total score of a reply is computed by summing up the scores in all the four categories. Thus, a reply gets a score between 0-4. The total score of an entire conversation is computed by taking the average of the scores of all the replies generated by the computer:

$$S = \frac{\sum_{i} Si}{N}$$

where Si = score of the reply i
N = total number of replies

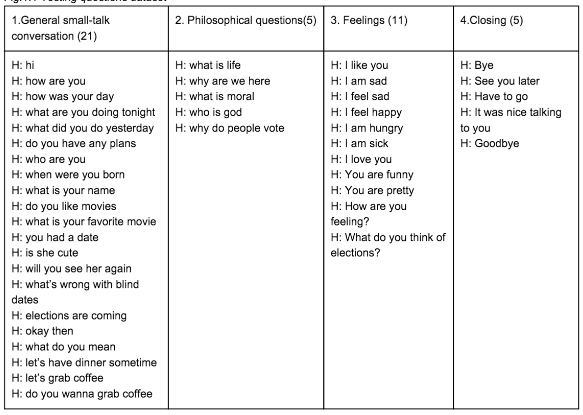
Besides being a score, GCT can also be viewed as a percentage of how many generated replies made sense. Thus, if we modify the formula for the score a little, we get another measure:

$$P = \frac{\sum_{i} Si}{4N} \cdot 100$$

where Si = score of the reply i, but P is now a measure of success how many generated replies made sense. Thus, it can be viewed as a quantitative metric of success of our model.

We have created our own set of questions (Fig.1.1) that targeted four different types of categories (General Small-Talk, Philosophical Questions, Feelings and Closing) and then tested and compared the results among three major categories of objects: available chatbots, different classifiers and datasets of different sizes.

Fig.1.1 Testing questions dataset



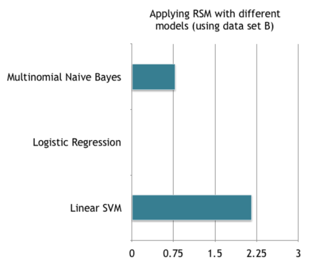
1.General small-talk conversation (21)	2. Philosophical questions(5)	3. Feelings (11)	4.Closing (5)
H: hi H: how are you H: how was your day H: what are you doing tonight H: what did you do yesterday H: do you have any plans H: who are you H: when were you born H: what is your name H: do you like movies H: what is your favorite movie H: you had a date H: is she cute H: will you see her again H: what's wrong with blind dates H: elections are coming H: okay then H: what do you mean H: let's have dinner sometime H: let's grab coffee H: do you wanna grab coffee	H: what is life H: why are we here H: what is moral H: who is god H: why do people vote	H: I like you H: I am sad H: I feel sad H: I feel happy H: I am hungry H: I am sick H: I love you H: You are funny H: You are pretty H: How are you feeling? H: What do you think of elections?	H: Bye H: See you later H: Have to go H: It was nice talking to you H: Goodbye

2. *Testing different classifiers*

We have tested Multinomial Naive Bayes, Logistic Regression and Linear Support Vector classifiers on Friends 1 dataset and computed both the in-category and overall score of the Granular Conversation Test. While Multinomial Naive Bayes didn't perform very well, it still performed better than Logistic Regression, that predicted <EOS> for every input. An explanation of this behavior can be found in the section above where we describe classifiers. The scores are:

Fig 1.3. Comparison of in-category and overall GCT scores of different classifiers on Friends 1

Type of question/Chatbot	Multinomial Naive Bayes	Logistic Regression	Linear Support Vector
Small-Talk	S: 11/21 = 0.52	S: 0/21 = 0	S: 43/21 = 2.04
Philosophical	S: 3/5 = 0.6	S: 0/5 = 0	S: 10/5 = 2
Feelings	S: 12/11 = 1.09	S: 0/11 = 0	S: 29/11 = 2.63
Closing	S: 7/5 = 1.4	S: 0/5 = 0	S: 9/5 = 1.8
GCT Score	TS: 33/42 = 0.78	TS: 0/42 = 0	TS: 91/42 = 2.16



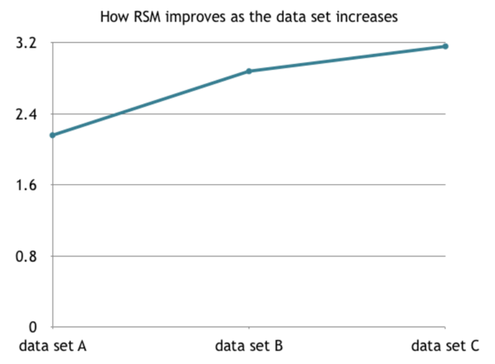
3.Testing datasets of different sizes

We have tested trainings datasets of different sizes from the Friends TV Show and computed both the in-category and overall score of the Granular Conversation Test. We have definitely seen an increase in our results by increasing the size of the training dataset. The scores that we obtained are:

Fig 1.4. Comparison of in-category and overall GCT scores of datasets of different size

Type of question/Chatbot	Friends 1 (Season 1)	Friends 1-2 (Season 1-2)	Friends 1-3 (Season 1-3)
Small-Talk	S: 43/21 = 2.04	S: 58/21 = 2.76	S: 67/21 = 3.2
Philosophical	S: 10/5 = 2	S: 12/5 = 2.4	S: 15/5 = 3
Feelings	S: 29/11 = 2.63	S: 36/11 = 3.27	S: 35/11 = 3.18
Closing	S: 9/5 = 1.8	S: 15/5 = 3	S: 19/5 = 3.8
GCT Score	TS: 91/42 = 2.16	TS: 121/42 = 2.88	TS: 133/42 = 3.24

S = GCT score in individual category; TS = total GCT score/conversation



where: data set A = Friends 1, data set B = Friends 1-2 and data set C = Friends 1-3

II. Qualitative

Here are some examples of conversations we were able to have with our conversational model:

Conversation 1	*Conversation 2*	*Conversation 3*	*Conversation 4*
H: who did you vote for? M: voted for obama H: now we'll have a new president M: but we have the same old problems A: do you want to be president M: but think about all the power A: what type of power B: they forget where they came from maybe you H: you should be president M: but run for office	H: what do you want M: you me H: why you me M: don't know it's just so mean to like be who he's not here yet so and sooooooo are you going H: this will not work M: is it out' of your league exits H: i can't do this M: oh we just delighted so uh you met yes someone huh H: you are too awkward M: so much H: you are so creepy M: oh much better than the coffee house	H: you had a date M: yes it was love at first sight H: was it a blind date M: you must be joking H: what's wrong with blind dates M: date someone don't even know H: okay then. you said it was love at first sight M: yes it was love at first sight H: that means you love her M: she's cute okay she's not ugly she's H: so she is not ugly M: she's just fat H: how fat M: she weighs 98 pounds okay okay	H: do you want to go out M: with me H: yes M: oh my god H: let's head out to grab coffee M: me yeah H: do you want to go somewhere M: c'mon H: are you having fun today M: yeah ok H: i have to pee M: i'll cancel the sodas

VI. Conclusion and Future Work

Using our Recurrent Statistical Model (RSM), we were able to build a simple and generalistic conversational model. Throughout the project, we were able to tackle the challenge of measuring how well a model works, which led to the development of our own test, the Granular Conversation Test (GCT), which aims at breaking down human conversation into four main components that make a conversation human. Using a Support Vector Machine (SVM) with a linear kernel was the approach that gave us the best result.

The other models we experimented with were able to capture other important aspects of human conversation, but these results came at high costs. The logic model successfully captured basic logic and was able to draw inferences from facts given, despite its obvious drawbacks since it's only able to parse and understand human utterances that follow a few hard-coded templates, and it's only able to respond with a limited set of inferences based on the enforcement of logic rules. The n-gram search model could capture long-range correlations between words but was extremely prone to rote learning/overfitting. The n-gram method was very limited, because it requires the user to enter the starting n-gram of the sentence and it can only generate responses for known starting n-gram.

RSM was able to generate basic conversations by extracting knowledge from noisy, relatively small and open-domain data. There are, however, drawbacks that result from the simplification of aspects of human communication, such as a consistent personality, general world knowledge, and human logic. Future improvements can help us capture more aspects of human communication and human nature. For personality, a new model could use multi-class sentiment analysis techniques together with RSM to create a coherent personality. Our logic model, which is able to make basic inferences based on logic rules, could be combined with RSM to build a system that models both human logic and human communication. Both combinations could model and simplify important aspects of human nature.

References

1. Turing, A. M. Computing machinery and intelligence. Mind, pp. 433–460, 1950.
2. “Automatic Capacity Tuning of Very Large VC-dimension Classifiers” I Guyon, B Boser, V Vapnik - Advances in neural information processing 1993,
3. “Support-vector networks” C. Cortes, V. Vapnik, Machine Leaming, 20, 273-297 (1995)