Case Study: Chat Categorization

Yu-Cian hong

Outline

Goal
Build AI chatbot for costumer assistance.

Task
Sort conversations that belong to different products, to build a training set for the chatbot.

Data processing

1. Preprocess texts
Text cleaning, tokenization, lemmatization, etc.

2. Gather labels
Get a complete set of product and service labels.

3. Label data
Active learning: hand-label a subset of chat entries, and repeat the process after each model training iteration.

4. Intent classification
Beyond product labels, conversations are classified into different intents, such as casual and business intents. Business intents are then further classified into different products and services.

Model training

Train model
Perform *intent classification* on each chat entry at each active learning iteration.

Goal

This project will attempt to build the training corpus for customer service AI chatbot. Digitized customer services have a lot of advantages over human customer service. It can provide efficient real-time customer assistance, because customers will not have to wait in line for available agents and can obtain assistance outside of business hours or when they are overseas. It can also help the business save a huge amount of human resource spendings on customer service.

Task

This project aims to link each chat entry to the right ancestor. Every time an agent get its turn to speak, it creates a new chat entry, and when another agent starts speaking, another new chat entry is created. In real world conversations, multiple threads on different subject matters often run in parallel. For example, before the customer service agent gets back to the customer about the information of a product, the customer may start an inquiry about another product or service. The conversation is on two different products in unsorted order. Therefore, the training corpus for the chatbot is unsorted. To find out the correct ancestor to link to for a chat entry, we can classify the utterance to the product it belongs to. We can then sort the conversations into separate threads that belong to a single product or service. Here I propose using a combination of named entity recognition and inquiry-response queue to sort the conversations.

Data processing

1. Text preprocessing
The corpus will go through standard text preprocessing. Non-English letters will be removed, except for question marks. The reason for keeping the question marks will be explained in business intent classification. The sentences will be turned into individual word tokens. Letters will be turned to lower case. Stopwords will not be removed in the preliminary model, and will be removed in the retrained model, in order to evaluate the impact of keeping a complete sentence structure. Light lemmaitization will be performed so that the POS tag of a word does not change.

2. Gather labels
To classify which product or service a chat entry links to, firstly there needs to be a complete set of labels that describe products and services. The labels can carry sub-labels. For example, a label can contain the product category and the specific product ID number, like “retirement plan A1”. To gather all possible labels, it is fastest to enquire documentation from the product and customer support department. It may be necessary to read some conversations to create more labels. Topic modeling techniques such as LDA can help extract labels and provide the word distribution for each topic. However, this will require sorting the conversations first, so this can work only on a subset of hand sorted conversations.

3. Label data — active learning
Because it is expensive to hand label all the conversation by human, labels will be created using active learning. A subset of data will be labeled with their intent, respective product

labels, and POS tags by hand. The initial subset will be used as the seed training data. After the first iteration of model training, we can come back to the unlabeled data, and choose to label some more label data points that can help gain the most information. That is, choosing data points that the model is most unsure of. There are different ways to sample the data. We can choose data with the highest entropy (ie. broader probability distribution among different classes.), or data with the lowest difference between the highest two probabilities, or if the most probable class of the data is lowest among all data points. When there're a large number of classes, choosing data based on the highest one or two probable classes is probably a better approach, because the probability distributions may be very broad for all data points. After new data are chosen, they are included in the next iteration of model training. This active learning iterations go on until the desired model performance is reached, or if model performance saturates.

4. Intent classification

1. Business intent (Sequence modeling + Inquiry-response queue)
   - product inquiry
   - response
     - explanation
     - affirmation

2. Casual intent (N-gram)
   - greeting
     - Opening
     - Conclusion
     - Other

Each sentence or chat entry has an intent. It can be a product inquiry, a response to an inquiry, or a greeting. There are two broad categories of intents in the context of this project - business intent and casual intent. Business intents include product inquiry and response, and casual intents include greeting.

Casual intent

A casual intent can be the opening speech (eg. "Hello"), close remarks (eg. "Thank you for being our valued customer."), or others that doesn't contain information or relevance to the product service (eg. "Thank you." "One moment please.") Casual intents are easier to implement by the AI chatbot, so the chatbot will not have to learn it. For example, opening and concluding remarks of a conversation are easy to formulate. Greetings in the middle of the conversations are usually not functional but only for politeness, so can be omitted. Casual can be classified using a simpler model than business intent classification, just to exclude them from the training model for the chatbot.

Business intent

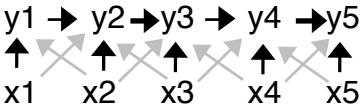
Business intents include inquiries related to a product or service, and responses that offers action, explanation, or affirmation to those inquiries. Such intents can be identified with named entity recognition models. Here I also propose the inquiry-response queue technique for identifying tags on the speech when named entity recognition has too limited information to perform the classification properly.

Model

Sequence models

Sequence models are superior for named entity recognitions than bag of words or keyword list approaches, because in real world NLP, input features are dependent on each other. It also has the advantage of clarifying ambiguities in tags because it has the surrounding context. There are several common sequence models, including the hidden Markov model, the max entropy Markov model, and the model of our choice — conditional random fields.

The hidden Markov model is the most simplified among the three. It assumes the input data are conditionally independent, and the probability of a data label is only dependent on the previous label. Here it is worth trying if a baseline model is needed for comparison. The max entropy Markov model (MEMM), on the other hand, has dependencies between adjacent labels and the full input sequence before the around the surrounding inputs. And the dependencies is directed. For example, in the figure below, y3 depends on y2 and x2, x3, x4, and because y2 depends on y1 and x1, x2, x3. However, this creates a problem called label bias. In plain words, observations farther down the line doesn't affect previous labels.



It is also reflected in the mathematical formulation of the conditional probability of MEMM below. The normalizer Z is a localized one.

$$P(y_i|y_{i-1}, x) = \frac{1}{Z(y_{i-1}, x)} exp\Big(\sum_{i=1}^{n} w_i f_i(y_i, y_{i-1}, x)\Big)$$

Conditional random field is superior to MEMM because it solves the label bias problem by using a global normalizer Z that sums over the full data sequence.

$$P_w(y|x) = \frac{1}{Z_w(x)} exp\Big(\sum_{j=1}^{n} \sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\Big)$$

with Z(x) defined as:

$$Z_w(x) = \sum_{y \in Y} exp\Big(\sum_{j=1}^{n} \sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\Big)$$

Therefore, our choice of model here for named entity recognition will be the conditional random fields. Besides the training data itself, documentation on products and services will be important supplemental corpus for named entity recognition.

Business intent classification

Before the model training starts, we will remove chat entries with casual intents. For identifying casual intents, we perform classification using a keyword list and N-gram. For classifying business intents, we perform (1) named entity recognition with sequence modeling, preferably with conditional random fields (2) inquiry-response queue.

After (1), the chat entry will get a label on its corresponding product or service. Subsequently, if it doesn't get a label or if the confidence of the label is low, it will further be classified as an inquiry or a response, or both, in order to use inquiry-response queue to apply a label on the data. After all chat entries are labeled, they can then be sorted into different conversation groups, and the goal of this project is completed.

Because we adopt the active learning approach, after the model is trained, the next iteration starts. New data will be queried and labeled, and the model will be retrained. The iteration continues until desired accuracy is achieved or if the model accuracy saturates.

Ways to identify an inquiry include whether it is a question, a command, or if the customer named a specific product or service, or something the customer says right after the opening greeting. To identify a question, pre-trained or self-annotated corpus can be used. Useful features can form a vector that the machine learning model uses. Features can include 5WIH words, POS tagging, the position of verb, keywords, etc. Identifying commands can use a similar approach. If a chat entry is not inquiry, it is classified as a response, assuming most greetings are already excluded from the data. Responses are classified into three different sub-intents: action report, providing information, affirmation. If a data doesn't fall into those sub-categories, it is dismissed as greeting.

The inquiry-response queue is a queue that stores inquiry response pairs. When the end of the queue has an inquiry by person 1, a response by person 2 can go into the queue; if the label of the new data is an inquiry, it can't go into the queue. Alternatively, it will be added to another queue that has a response at its tail. Each product or service has their own queue. When an inquiry about a product comes up, the queue is created. To better illustrate how the inquiry-response queue works, we will go through an example conversation between a customer and an agent about two products. We will see how the queues evolve through each chat entry. A full conversation without intertwining explanations is in the appendix.

Example conversation:

1. Customer: Hello. (Greeting - opening)

2. Agent: Hello, how can I help you? (Greeting - opening)

3. Customer: What is the price of product X 35? (Product X 35 opened - customer.inquiry)

Here is a new inquiry about the product X 35, so its queue is created.

QueueX35 = [customer.inquiry]

4. Agent: I see that you would like to know the price of product X 35. I'm happy to help look it up for you. (Greeting - X35)

5. Customer: Thanks. (Greeting)

6. Agent: Please give me a moment. (Greeting)

7. Customer: Can I also ask you about product Y 61? (Product Y 61opened - customer.inquiry)

Here is a new inquiry about the product X 35, so its queue is created.

QueueX35 = [customer.inquiry]
QueueY61 = [customer.inquiry]

8. Agent: The price of product X 35 is $200 / month. (agent.response - X35)

Here is an agent response classified as X35, so it goes into QueueX35.

QueueX35 = [customer.inquiry, agent.response]
QueueY61 = [customer.inquiry]

9. Agent: Sure! One moment while I look it up for you. (Greeting - Y 61)

10. Customer: I see. I'd like to purchase it. (greeting + customer.inquiry)

Here is an unnamed customer inquiry that has to little information for named entity recognition. Because QueueY61 already has a customer inquiry that is waiting for agent.response, it will not take this entry. Therefore, this customer inquiry will instead go into QueueX35 and be labeled as X35.

QueueX35 = [customer.inquiry, agent.response, customer.inquiry]
QueueY61 = [customer.inquiry]

11. Agent: Sure thing. I will process the transaction for you. (Greeting)

12. Agent: And the price of Y 61 is $3000/ year. Would you like to purchase it? (agent.response + agent.inquiry - Y61)

This is clearly about product Y61, so it will be fed into QueueY61, as a dual intent entry.

QueueX35 = [customer.inquiry, agent.response, customer.inquiry]
QueueY61 = [customer.inquiry, agent.response + agent.inquiry]

13. Customer: I'd like to purchase it.

This is an unnamed customer.response.affirmation, it needs to pair up with an agent.inquiry, so it goes into QueueY61. It is also an inquiry that asks the agent to perform an action, so it is added into QueueY61 as a dual intent.

QueueX35 = [customer.inquiry, agent.response, customer.inquiry]
QueueY61 = [customer.inquiry, agent.response + agent.inquiry, customer.response + customer.inquiry]

14. Agent: Sure. I'll process that for you in a moment. (Greeting)

15. Agent: Your transaction of X 35 is successful. You will get an email confirmation in a moment. (agent.response - X35)

This is an agent response reporting its action, clearly on product X35. It will go into QueueX35.

QueueX35 = [customer.inquiry, agent.response, customer.inquiry, agent.response]
QueueY61 = [customer.inquiry, agent.response + agent.inquiry, customer.response + customer.inquiry]

16. Customer: Ok.Thanks! (Greeting)

17. Agent: Your purchase of Y 61 is completed. Is there anything I can help you with today?

This is clearly agent.response on product Y61, so it goes into QueueY61.

QueueX35 = [customer.inquiry, agent.response, customer.inquiry, agent.response]
QueueY61 = [customer.inquiry, agent.response + agent.inquiry, customer.response + customer.inquiry, agent.response]

18. Customer: It's all good. Thank you for all the help! (Greeting)

19. Agent: Thank you for being a customer of Citi. I hope you have a wonderful day! (Greeting)

The inquiry-response queue model presented through this example works best with conversations that contain exactly two different products or services. This is our best assumption because in the real world, it is more common to have parallel conversations on two different subject matters than to have more than two. In the example, one chat entry only contains one product / service. In the case an entry contains two products, it will duplicate itself and go into both queues. There are some grey areas whether a chat entry should be classified as greeting or response. Detailed sub-categories of responses can help clarify the grey area. For example, a response as affirmation from an agent should be considered as greeting (eg. Agent: sure I will process that for you in a moment.).

Conclusion

This project aims to sort conversations on different subject matters into respective ordered threads. We projected this as a sub-project of a customer service AI. Sorted conversations can be good learning corpuses for the AI chatbot. We adopt a semi-supervised training model for sorting conversations, because hand labeling all chat entries are expensive. We choose an optimal subset of data to hand-label using active learning methods, and iterate model training with newly labeled data. For identifying relevant product from a conversation, we adopt a sequence modeling approach to perform named entity recognition. We choose conditional random fields for its superior performance over other Markov models. Detailed intent classification also assisted in cleaning the data — only functional conversation with business intent is used for training. Classifying business intents into inquiry and response creates an opportunity for a composite model. Where the named entity recognition can’t identify a label, the inquiry-response queue implements a rule that helps identify the label.

Appendix - full example conversation

1. Customer: Hello. (Greeting - opening)
2. Agent: Hello, how can I help you? (Greeting - opening)
3. Customer: What is the price of product X 35? (Product X 35 opened - customer.inquiry)
4. Agent: I see that you would like to know the price of product X 35. I'm happy to help look it up for you. (Greeting - X35)
5. Customer: Thanks. (Greeting)
6. Agent: Please give me a moment. (Greeting)
7. Customer: Can I also ask you about product Y61? (Product Y 61opened - customer.inquiry)
8. Agent: The price of product X35 is $200 / month. (agent.response - X35)
9. Agent: Sure! One moment while I look it up for you. (Greeting - Y 61)
10. Customer: I see. I'd like to purchase it. (greeting + customer.inquiry)
11. Agent: Sure thing. I will process the transaction for you. (Greeting)
12. Agent: And the price of Y 61 is $300/ year. Would you like to purchase it? (agent.response + agent.inquiry - Y61)
13. Customer: I'd like to purchase it. (customer.response + customer.inquiry - Y61)
14. Agent: Sure. I'll process that for you in a moment. (Greeting)
15. Agent: Your transaction of X 35 is successful. You will get an email confirmation in a moment. (agent.response - X35)
16. Customer: Ok.Thanks! (Greeting)
17. Agent: Your purchase of Y61 is completed. Is there anything I can help you with today? (agent.response - Y61 + Greeting - closing)
18. Customer: It's all good. Thank you for all the help! (Greeting - closing)
19. Agent: Thank you for being a customer of Citi. I hope you have a wonderful day! (Greeting - closing)