A German Dialogue Dataset for Tech Support
COMP 551
Applied Machine Learning

Thomas Page — thomas.page@mail.mcgill.ca — 260771672
David Tamrazov — david.tamrazov@mail.mcgill.ca — 260561439
Alexander Wong — alexander.wong4@mail.mcgill.ca — 260602944

September 27, 2017

URL: $https://github.com/David-Tamrazov/dialogue\_dataset\_curator/tree/master/final\_data$

1 Overview

Our dataset is gathered from Apple Support Community discussion boards, and would be useful for training machine learning systems specifically for the task of tech support in German. The applicability of this dataset is definitely most geared towards supporting Apple-related questions, but probably can be generalized to most technical customer service platforms.

2 Dataset Description

The dataset is made up of posts and responses from the German Apple Support Communities. The corpus includes a variety of formats such as block quotes, links, lists, and conversational dialogue between the questioner and the respondents. The data was collected over a large range of discussion posts, approximately 13000, to ensure that many of Apple's products and services were covered. Most of the individual dialogue pieces consist of a question with multiple responses from various users of the Apple Support Communities.

To add to the usefulness of the dataset, we extracted **score**, a feature beyond the project specifications. Score (or the number of likes for each post), reflects the helpfulness of a response, or in the case of the he original poster (OP), whether someone else had the same question). This dataset contains most of the same conversations as the set of 13000, except smaller with only 6000 conversations. For the sake of the assignment we are submitting two datasets:

- $NotoriousApplePickers\_GER.xml$
- $NotoriousApplePickers\_GER\_Scores.xml$

3 Discussion

Methodology Our goal in building our dialogue curator was to provide a dataset large enough in magnitude, and diverse enough in content, to be used in training an AI responsible for automating customer support in Apple?s German discussion boards. To that end we employed the Python library BeautifulSoup4 to scrape the html data from the discussion boards, and parse the data for information on the thread that we could use to build our dialogues, such as usernames in a topic's discussion, and the text of the responses themselves. The benefit to scraping conversations from a website run on individual user input is the colloquialism in the responses. In contrast, a dataset developed from a formal setting like a FAQ would have a different type of language that would cause the automated tech support to sound robotic.

The generic nature of Apple's discussion board URLs (https://communities.apple.com/de/message/xxx where xxx = iterated integer ID) allowed us to not only cover a wide breadth of topics, but also meant we could easily loop through a large range of URLs (13000 in our case) and build our dataset as we parsed them one by one. This kept our `apple_dialogue_curator.py` script simple, readable, easy to optimize, and easy to maintain.
For the data analysis, we used the Python library ElementTree to easily parse through our dataset.

Defining a Conversation One of the key design decisions made was how were we going to define a conversation and therefore what discussion threads would make the cut for the dataset? Our initial intuition was to only include discussions with at least one instance of a response from the OP, in addition to their original post. We then revised this to at least one instance of non-consecutive posts by any user, for the inclusion of the OP does not preclude a thread from being a conversation among others.

However this too proved to be unsatisfactory; in the end, we settled on the current definition of a conversation, which simply includes any thread with at least one response.

The reason behind this choice was two fold. First, threads with a single response allowed us to incorporate topics that could be swiftly, and succinctly concluded. It is our intuition that keeping a simpler dataset would ultimately result in a simpler, more accurate customer support model. Thus if we included datasets with at least one response, this would then incorporate all threads that had been closed in exactly one response (implying a simple problem-solution dialogue). This then feeds into the second factor behind our decision, which is that we believe the best tech support takes the form of a singular, concise response (complexity and obscurity of the original question aside). As such, we sought to provide any machine training off of our dataset with as many examples of this caliber as possible, to help guide its weights and hyper parameters in that direction.

Other Design Choices To fit our the scraped data to the requisite .xml format, we made minor modifications to the extracted text. We changed all instances of '<' and '>' to '_open_angle_' and '_close_angle_', because they clearly would interfere with parsing xml. While parsing our corpus for basic analysis we found that the ampersand and at symbols would also provide unwanted interference, so changed them to '_and_' and _at_' respectively.

Data Characteristics and Analysis While the responses are technical in nature, the benefit to scraping conversations from a website run on individual user input is the colloquialism in the responses. In contrast, a dataset developed from a formal setting like a FAQ would have a different type of language that would cause the automated tech support to sound robotic.

Our corpus of 13000 conversations has an average of 6.22 utterances per conversation, with the longest conversation consisting of 16 utterances. It also has an average of 2.65 users contributing to the post, with a maximum of 14 distinct users contributing to a single thread.

Analysis of the corpus including scores reveals an average of 0.33 posts with at least one like, revealing a higher than random approval rate and only a maximum of 4 posts with at least one like in any conversation, which if anything does not speak well for the Apple support community. Meanwhile, the average OP score (other users with the same question) is 3.75, with a max of 1243.

Area for Improvement One important thing we missed out on was proofing the data after scraping, specifically to ensure there were no repeated threads. While this is impossible in theory, human error in our code could still allow for this. While it may be inconsequential, such repetitions would taint the quality of the corpus.

We also could have scraped using different definitions for a conversation, to determine which would be more helpful.

Final Remarks Suggestions we have for future users of the dataset are to filter the data to meet the your training needs; there are many ways to classify these tech support threads, for helpfulness or clarity etc. Training on the full corpus may not produce the most helpful of tech support agent.

4 Statement of Contributions

Thomas got the ball rolling by introducing us to the BeautifulSoup Python library that played a large roll in simplifying the task from the coding side. He also wrote the initial scripts to test feasibility, and suggested sourcing our data from Apple tech support pages. Upon agreeing on our objective, we then worked cooperatively to create a program that extracted the data we wanted from the pages, without double counting or skipping posts. Dave and Alex worked on optimizing Thomas' initial program, and implemented the logic for testing each discussion thread to meet our definition of a conversation.

Thomas ran the program from home to scrape for the main assignment corpus (approximately 7 hrs), Alex did the same for the corpus with scores (5 hrs

because fewer examples), and all three of us did some data analytics to include in the report. Alex wrote the first draft of the report, and from there everyone contributed to finalizing it for submission.

We hereby state that all the work presented in this report is that of the authors.