Building CMU Magnus from User Feedback

Shrimai Prabhumoye*, Fadi Botros*, Khyathi Chandu*, Samridhi Choudhary*, Esha Keni*
Chaitanya Malaviya*, Thomas Manzini*, Rama Pasumarthi*, Shivani Poddar*
Abhilasha Ravichander*, Zhou Yu, Alan Black
Language Technologies Institute, Carnegie Mellon University
{sprabhum, fbotros, kchandu, sschoudh, ekeni, cmalaviy, tmanzini,
rpasumar, spoddar2, abhilasha, zhouyu, awb }@cs.cmu.edu

Abstract

Recent years have seen a surge in consumer usage of spoken dialog systems, due to the popularity of voice assistants. While these systems are capable of answering factual questions or executing basic tasks, they do not yet have the capability to hold multi-turn conversations. The Alexa Prize challenge provides us a great opportunity to explore various approaches and dialog strategies for building a multi-turn conversational agent. In this report we identify key challenges to build a social conversational dialog system, and present CMU Magnus, an intelligent interactive spoken dialog system that can hold conversations over a range of topics. The system learns and updates itself over time, and can handle argumentative or subjective conversations.

1 Introduction

Spoken dialogue systems have transformed how humans engage with technology in the present era. Socialbots, that are able to coherently engage with the users are envisioned to be the harbingers of this transformation. Our aim for this work was to develop one such social bot, that is adept in conversing with the users on popular topics in an engaging and coherent fashion. Our bot, CMU Magnus, has been resultant of three major milestones (stages) as detailed by means of sections 2, 3 and 4.

Stage 1 is fundamentally a knowledge assimilation system with different modules (listed below) representing the various facets of a conversation. While these modules were independently adept at engaging the user for a single turn, they lacked the capacity to engage across multiple turns. Independence in the modules resulted in the system being unable to capture user context and coherency, thus making the conversation sound fragmented. To address this problem, we worked on a Finite State Transducer (FST) architecture that would selectively query each of these modules for assimilating knowledge while maintaining user context and conversation state. Thus, Stage 2 of CMU Magnus was introducing a generic FST module which has been detailed in section 3.1. Novel aspects of this approach were introducing domain independent templates of agent utterance conditioned on the state of the conversation. These states were a part of a hierarchal ontology of states that were activated based on a range of user utterances. Introducing this module increased our initial ratings suggesting that the user engagement in the system improved. Finally, the Stage 3 of the system involved extending the FST architecture to include other domains such as sports, travel and a Game of Thrones (TV Series). This approach is contrasted with the existing techniques because of it's domain independent module architecture, that can eventually be trained using techniques such as reinforcement learning.

*All authors contributed equally to this article.

1st Proceedings of Alexa Prize (Alexa Prize 2017).

2 Stage 1

In the first stage of developing Magnus, we created multiple modules to cater to different intents. These modules are based on the following categories: opinions, factoid questions, non-factoid questions, news and Magnus's story. In this stage, the input query is passed to each module. Each module then returns its response to the query along with a confidence score. The response with the highest confidence score is returned by the dialog manager to the user.

2.1 Opinion Mining System

The aim of the opinion mining system is to be able to respond to questions that are opinion seeking in nature, using a retrieval based approach over comments on news articles to identify human-generated comments of positive and negative polarity. We used Washington Post news articles and the comments, scraped using the Washington Post API. We briefly describe the system architecture below.

Tokenizer and Entity Recognition: The user query is tokenized and passed through an entity recognition module to identify entities (person names, locations, etc.). Instead of passing all the tokens or the words in the query to the indexed corpus, we pass this filtered set of entities for targeted and more relevant retrieval of opinions. If the tagger fails to identify the entities from the tokens, the raw set of tokens are processed to remove the stop words and then passed to the corpus for retrieval.
Sentiment and Relevance: The comments across all the articles in the dataset were annotated with sentiment values (Positive, Negative, Neutral) using a lexical based sentiment analyzer from [5]. These sentiment valence scores are used to generate opinions on the entities recognized from the user query. Lucene was used for indexing and relevance was computed as a combination of high sentiment score and high query-document similarity computed using BM25 score.
Maximal Marginal Relevance (MMR) Reranker: The indexer returns a list of most relevant comments for the query passed to the system. This list is further divided into 3 separate lists, one for each sentiment value (Positive, Negative and Neutral). Each list is sorted in decreasing order of their sentiment scores. The sorted lists are passed to the MMR [3] reranker. Each sentiment annotated list of comments are seeded with their original BM25 relevance score that gets adjusted and re-sorted by this module.
Opinionated Extractive Summarization: The output from the reranker produces multiple opinions for each opinion valence (Positive, Negative or Neutral). These multiple opinions are passed to the summarization module in order to get an informative, well formed opinion from multiple opinions. This is done as an extractive summarization, with an approach based on TextRank [7], which identifies the set of words or phrases that are most informative from a given document. A graph of words/phrases is constructed, and centrality over this graph is used to determine the most important words/phrases, which are used to construct the summary. We build the summary based on subset of opinions of a certain valence positive/negative/neutral) as specified by the user.

2.2 Community Question Answering System

The Community Question Answering module (CQA) was designed to be an open-domain question-answering system, primarily used to respond to non-factoid and abstract questions (For example, *My cat is pregnant. What should I do?*). The CQA module is a retrieval model. We aimed to build a system that could satisfy the below mentioned properties: *Coherence*: The responses should be comprehensible to the user, *Relevance*: The response should be related to the user's utterance and not totally off-topic, *Response Time* : A bot that is slow to answer can quickly disengage or bore the user.

In the following sections we discuss the structure of our dataset, the strategies we followed to use this data in a dialogue system and present initial results.

2.2.1 Data Sources

For this module, we relied on the Yahoo L6 dataset [1], hence referred to as L6, as the source of all of our conversational data. This dataset contained approximately 4 million questions as well as categories for the different questions such as entertainment, relationships, sports etc. The structure of any typical question-answer thread includes: *Question Title*, which usually includes the main question being asked, *Question Content*,where the asker can specify any additional information they

choose, *Best Answer*, the user has the option to select one of the answers as the best answer, *Answer Posts*, answers posted by community users in response to the question.

2.2.2 Approach

Searching or scraping answers at run-time increases the response time of the system. To decrease the response time, we built an index over this dataset. We used the Lucene search engine and constructed our own search index where each entry in the index contained several fields: question title, question additional information, answer, answer rank. Hence, a new entry in the index was constructed for every answer to ensure that we were able to rank the answers in terms of what the community had decided was the best one.

We find that questions from users in the real-world are very diverse and we are often not able to find a semantically equivalent question in our dataset. Thus, when we do not identify a complete match to the question, we instead find questions that are in a similar domain and consider all the sentences in them. We then rank these based on a number of features (semantic coherence, BM25 similarity, reranker score, average synset path similarity score as defined by WordNet).

Neural Reranking: We experimented with a neural approach to reranking question matches[13, 11, 9] . We first collect the top-25 matches using Jaccard similarity as the similarity metric, to create a smaller bounded search space. We then apply a neural reranker to this 25-best list to select the question that is closest to the user query. We generate a representation of both the test query and the paraphrasing using a bidirectional-LSTM and use the loss function as specified:

$$L = max(0, M - cosine(q*, q+) + cosine(q*, q-))$$

where q* is the test query, q+ is a question that has the same meaning as q* and q- is a question that does not. In other words, we learn a embedding over each of the questions, and learn to recognize similarity using a maximum margin loss.

Maximum-Entropy Reranking and Summarization: Additionally, we experiment with Maximum Entropy Reranking of the answers using features such as normalized length, number of special characters, presence of words like 'LOL', normalized Okapi BM25 similarity to the question, averaged word2vec similarity and a loss function as shown.

$$L = \sum_i w^T f_i(y_i*) - log \sum_y exp(w^T f_i(y_i))$$

However, we found that this gave poor performance in terms of finding good answers, as opposed to the answer selected by the asker of the question in the L6 corpus. We also experiment with extractive summarization from multiple answers using MMR but found the response to be too incoherent to return to the user.

2.2.3 Results

The performance of the system varies drastically based on the domain of the question that is being asked. We conduct evaluation by performing human user evaluations over the course of one conversation. Three annotators rated the dialog system on the basis on a Likert scale from 1-7 on the basis of coherence, relevance and engagement. These results averaged together were **Relevance - 5.17**, **Coherence - 6** and **Engagement - 5.34**

The above results clearly show that nearly all evaluated aspects of the CQA system were able to outperform the neutral rating of 4. Empirically all these results makes intuitive sense, as the responses that were presented to the users were directly selected from the corpus as opposed to being generated statistically. As can be clearly observed from the quality of the answers in L6, users attempted to answer the questions posed in a relevant, coherent, and engaging manner. As such, the fact that our system achieved above neutral ratings on all of these metrics is not surprising. However, there are some clear and obvious drawbacks to our approach. Most notably are the failure cases. Questions that were posed to the system were either answered in a very satisfactory manner, or failed completely. There was rarely a response from the system that was would have been neutrally rated. Users either loved or hated the responses.

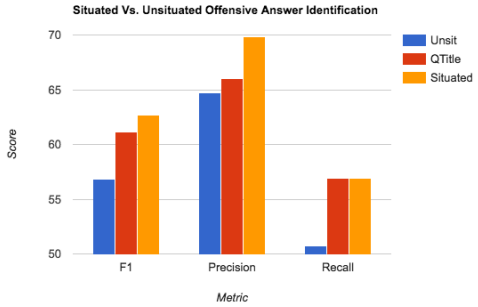


Figure 1: Performance of Unsituated vs. Situated Models for Offensive Content Prediction. Situating the answer substantially improves the performance of offensive content identification.

	Weights
	SVM
Negative Sentiment	0.87
no,	.519
!	.513
check	.45
asking	.43
BOL WHAT	.43
WP DT	.41
what a	.35

Table 1: Most influential features for non-offensive posts to be classified as offensive

2.3 Abusive Content Identification

We found that user-contributed answers on Community Question-Answering websites and comments on newspaper articles tend to be highly noisy and sometimes offensive. Most prior work in this field aims at identifying abuse from the perspective of a third-party [10, 8, 4], often using blacklists, regular expressions or features from the post to identify offensive content.

One of the biggest challenges in the area of identifying offense in online content is to first define 'offense', and what could possibly be offensive to the asker. For example, the question *'Tell me a blonde joke'* on Yahoo! Answers attracts several responses, most of which would be considered offensive from a third-party perspective but which are unlikely to be offensive to the asker of the question. We formalize the notion of offense by leveraging Levinson's politeness theory[2].

To detect offensive content, we scrape data from the Yahoo Webscope dataset which is likely to contain such offensive language. We sample using a seed lexicon with the intention of extracting threads which are likely to contain abuse. For each post in which any term in the seed lexicon appears, we extract the entire question thread and add it as a candidate. In this way, we extract 1775 question-answer threads with 19986 posts. We subsample a set of 390 answers to annotate for offense. The average inter-annotator agreement on this dataset is 0.645.

We propose a novel step towards more nuanced identification of offensive posts in Community Question-Answering websites, by also taking into consideration information from the question, as well any additional content provided by the asker. The primary contributions of this module are: ***C1***: A conceptualization of *offense* using politeness theory. ***C2***: Baseline methods on our work-in-progress corpus, annotated in accordance with the guidelines from the coding manual we developed.

2.3.1 Feature Space Design

We incorporate various considerations while trying to identify offense. The four basic categories of features are surface, syntactic, readability and sentiment-based features.

Surface Features. These include unigrams, bigrams, trigrams, within each post, as well as those of the question being asked. Additionally, we also consider length of the answer, type-to-token ratio, punctuation and character n-grams.
Syntactic Features. We incorporate POS tag features and Word-POST tag pairs to identify common structural markers of abusive posts.
Readability. We consider readability features to evaluate the hypothesis that posts on community question answering websites which are offensive, tend to be less articulate than posts which are genuine in their intent to answer the question.
Sentiment. We analyze the positive, negative and neutral sentiment valence for each post.
We use a simple SVM with a linear kernel as a baseline model. This helped us to learn a classifier

over the described features so as to predict whether a given answer was offensive or not. We perform ten-fold cross validation.

2.3.2 Experiments and Results

Analysis of the annotated data reveals the features most correlated with offensive content (excluding features related to the seed-lexicon) as shown in Table 1. Using the vanilla implementation of SVMs i.e. solely using answer features, we achieve a precision of 64.7% and a recall of 50.7% for offensive content identification, with an overall accuracy of 87.1%. We then move to a more informed version of this baseline, called Situated SVM. Using this approach we were able to account for question context while predicting offense. As part of our analysis we experimented with two versions of this approach. *First*, we use only the question subject as additional encodings over baseline SVM. *Second*, we experiment with using both the question subject and the content as additional encodings over the baseline SVM. As depicted by Figure 1, while both the variants beat the traditional SVM baseline, the latter clearly outperformed the former in all the metrics. Quantitatively, we were able to achieve a precision of 69.8% and a recall of 56.9% for offensive content identification, with an overall accuracy of 88.7%. Based on these experiments we observe that taking into account features representing the question and it's content leads to substantially better identification of offensive content rather than a ham-fisted approach which only takes into account the answer, without situating it.

We also analyze the specific features that contribute to offense detection (Table 1). We see that features like *you're*, *being a* are high indicators of abuse, potentially given their affiliation to what would mostly follow these words in a sentence. Similarly, words like *if* which usually might follow a reasoning pattern are more negatively correlated to offense. This aligns with our expectations as more rational answers would be less offensive as compared to emotional answers.

2.4 Neural Model

We trained a skipthought encoder-decoder on the OpenSubtitles [12] data. The skip-thoughts vectors model abstracts the skip-gram model to the sentence level. It encodes a sentence to predict the sentences around it. This model depends on having a training corpus of contiguous text. In this model, an encoder maps words to a sentence vector and a decoder is used to generate the surrounding sentences. In their framework, the authors [6] have used an Recurrent Neural Network (RNN) encoder with Gated Recurrent Unit (GRU) activations and an RNN decoder with a conditional GRU.

2.5 Backstory

This module maintains the story of Magnus to answer basic questions the users may ask. For the implementation of this module, we maintain a dictionary of related questions. Each of these questions can have multiple answers to avoid repetition. We match the incoming query to a question in the dictionary using two methods - Jaccard scores and Google pretrained word vectors (by taking an average of all the word-embeddings in that sentence).

2.6 News Module

We used the Washington Post News Articles to build a Q&A system that would leverage heuristic rules to situate a question, and subsequently return it's answer. We filtered on the types of questions such as *Whats* to retrieve inanimate identifiers, *Whys* to retrieve reasoning based answers, *Whens* to retrieve locations and time stamps and so on. The answers given by this module were highly structured excerpts from news articles. These responses were not conversational.

2.7 Evi - Factoid Module

We have used the Evi API provided by Amazon to answer factual questions and as a fallback module for our system.

2.8 Issues

Context. One of the major drawbacks of this approach is that context is not maintained while drawing responses from multiple systems.

Module	Avg Accuracy	Macro Recall	Macro Precision	Macro F1
Backstory	74.45	61.37	62.18	61.47
Evi	93.58	89.48	89.82	89.36
Neural	58.91	58.90	58.98	58.81
News	74.74	55.22	54.63	54.77
Opinion	65.19	61.87	61.20	61.33
Yahoo	71.31	67.04	67.70	66.86

Table 2: Cross Validated Classification accuracies for each module

ASR errors. We noticed some ASR errors related to content words of the user query. Manual inspection showed that even though a few content words are identified wrongly by the ASR, the over-all sentiment of the input query remains intact. We use this observation in the next stage of development of the system.

Classification. Each query was passed to each of the modules mentioned above and a response was generated. Hence, we have 5 responses for each query. Each of these query-response pairs was manually annotated for binary labels - appropriate or inappropriate. We manually annotated 700 such query-response pairs. We built a binary classifier for appropriateness for each of the modules mentioned in Section 1 based on the manual annotations. The precision and F1 scores for each of the modules is shown in Table 2. We used unigrams, bigrams, presence of Wh words etc as our feature set for classifier. Due to unavailability of large annotated data tuned for our modules, we performed a five fold cross validation to assess the performance of our system. This was subsequently used to rank the responses of the different modules while interacting with the user.

Coreference Resolution. In this part of work, we used straight forward approached to add some basic context-maintenance features to the system. We manage to perform some rudimentary context maintenance and realize the challenges in intricately webbed context in a conversation.

One basic method of incorporating coreference resolution is to run a state-of-the-art coreference system over the current user utterance as well as all utterances from previous turns and replace all co-referent mentions with the phrases they are linked to. We use the Stanford Deterministic Coreference resolution system for this task. We observe that the Stanford coreference resolution system resolves third person singular pronouns and some noun phrases correctly, but does not do a good job at resolving 'that', 'this', 'there' and split antecedents. We also notice that it resolves first person and second person pronouns, such as 'you' and 'me', which would be detrimental for a dialog system. Based on our observations, we make some heuristic extensions to the system.

We **ignore the resolution of split antecedents**. We incorporate the following strategies on top of the Stanford system to aid in the resolution of some mentions which are not resolved by the system:

- If 'this' and 'that' occur in a sentence and if they do not occur as an adjective, then replace with 'it' so that it can be resolved by the Stanford system.
- If 'there' occurs in a sentence, then find the most recently mentioned location within the previous few utterances and substitute with that location.
- Do not resolve first and second person pronouns.
- Do not replace any phrases with pronouns.
- Store the person and location named-entities so that we can retrieve the most recent entity for resolving 'when' and 'who' questions. We can also use the most recent location entity for resolving 'there'.

The following are a few examples of conversations in which context maintenance by resolving coreference using the Stanford coreference system and our strategies works well:

However, we observe that coreference resolution does not completely solve the problem of context maintenance. For example, consider the following conversation:

Original: Alexa, Play Cheap Thrills by Sia, **Modified**: Alexa, Play Cheap Thrills by Sia
Original: Alexa, Play the entire album, **Modified**: Alexa, Play the entire album

In this conversation, we can see clearly that “play the entire album" should have been resolved to “Play the entire album by Sia which contains the song Cheap Thrills". Coreference resolution, as we see, is not able to resolve and maintain this information. Using coreference resolution and storing the entities mentioned by the user, we can perform context maintenance in a completely domain-agnostic way. This type of context maintenance can maintain coherence to some extent, but it cannot talk/remember specific information about entities. For example, this system may remember that the user mentioned "France", but it will not store the user’s opinion about France. Talking or remembering specific facts about entities requires the system to identify which aspects of entities are interesting, which is very difficult in an open domain.

3 Stage 2

To address the issues identified in Stage 1, we created a finite state transducer (FST). FST makes it easier to maintain context about a topic in the conversation. It also enforces coherence in the conversation between different turns.

3.1 Finite State Tranducers for Conversations (FST)

We have approached a conversation as a combination of serveral topical FSTs. Herein, we envisioned a user to switch both inter and intra FST. The first FST we worked on pertained to ’Movies’. An FST, how we constructed it spanned across 2 facets. The state of a **user** and that of a **conversation**. While the states of the user were capturing user context such as *Has the user watched this movie*, *Have we conversed with this user about this movie in a prior session* and so on. The conversation states on the other hand listed the state that the conversation itself was in, for instance *Introducing a movie/topic*, *Finding that the user is disengaged* and so on. The FST’s state transitions were thus informed both by the content as well as the sentiment of user utterances. We had a series of intent classifiers that would both ascertain the content as well as the sentiment of the user reply. For eg: In the state of ‘INTRO_MOVIE’, the system introduces a movie and asks the user if he/she has watched the movie. The system utterance in this state is ‘Arrival. It was such an engaging movie. I really enjoyed it. Have you watched it?’. The user can either respond with a positive sentiment of watching the movie or negative sentiment of not watching the movie. Depending on the sentiment of the user utterance, the FST goes to the next state of ‘QUERY_WHAT’ or ‘QUERY_WHY’. If the user has watched the movie then the system transitions in the ‘QUERY_WHAT’ state where the system asks the user what he/she liked about the movie. If the user has not watched the movie then the system transitions in the ‘QUERY_WHY’ state where the system inquires why the user has not watched the movie. The underlying system architecture of an FST module has been depicted by means of Figure 5. We further experimented with 2 facets of state to state transitions as suggested in the subsections below.

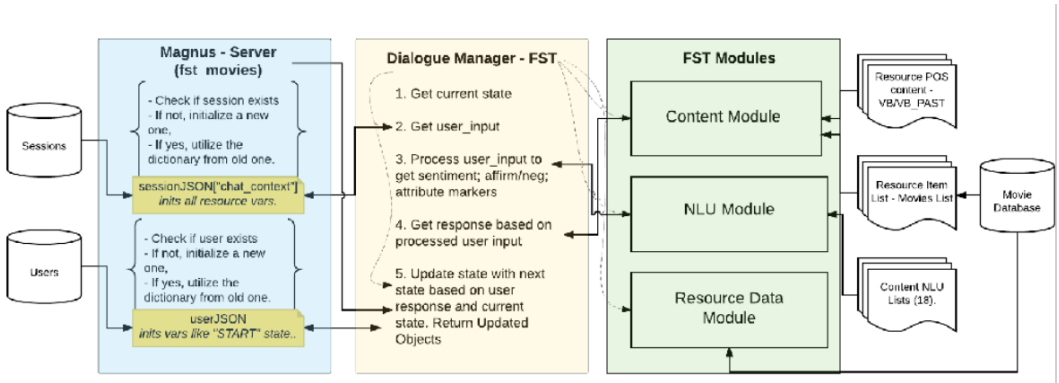


Figure 2: System Architecture of a baseline FST Module.

3.1.1 Deterministic Transitions

We started off with specifying the next state for the user, conditioning the content of this state on the user reply. This would mean that no matter what the user said for eg. *change the topic* an FST after

introducing a movie would always ask the user about the same movie. This would work well if the user was into the conversation, but provided very less flexibility for the conversation to go anywhere else.

3.1.2 Non-Deterministic Transitions

Thus, to tackle the problem of making the conversation flow more organically, we introduced non-deterministic state transitions. Herein, we initially gave high probabilities to states that were more likely to occur next, but we chose a range of states first based on the user input, and then further conditioned the replies within these states based on the content and sentiment of the user utterance.

We have over 20 states in this FST ranging from introductory states to states such as "QUERY_WHAT", "QUERY_WHAT_LIKE", "QUERY_WHY" and so on. While the initial deterministic transitional model attempted to align users to a specific set of states, eventually we adopted a more flexible and preemptive strategy wherein, we would exit the user out of one topic when detected continued disengagement.

It is important to note here that this base FST was built such that it was easy to leverage the exact same architecture for other topics as well. The data source for movies was IMDB. Thus, To create a FST for another topic, we only needed to modify the system utterances to match the topic and change the data source.

3.2 Issues

One of the major drawbacks of this model is that it is constrained to one topic restricting the flow of the conversation. At times, the user conveys intents that are not captured by our FST model. For example, after the 'INTRO_MOVIE' state, the user may say 'No, I have not watched the movie. What is it about?'. In this case, the FST identifies the negative sentiment of the utterance and transitions to the 'QUERY_WHY' state where the system asks the user 'Why not? Do you not like such genres?'. Hence, the intent conveyed by the user is lost and the FST responds inappropriately.

4 Stage 3

In an attempt to fix the limited conversational scope that was present with a single FST we expanded Magnus to handle multiple FSTs. This allowed Magnus to speak about a variaty of topics while still engaging in coherent and strucutred conversation.

4.1 Multiple FSTs

FSTs are constrained to one topic. To be able to talk about multiple topics, we built multiple FSTs catering to some of the topics. We have SPORTS_FST which talks about the upcoming football games. The TRAVEL_FST talks about the various cities visited by Magnus, what are the exciting things to do in these cities, the weather in the city, the shopping destinations and the local cuisine of the city. It also indicates how safe the city is to visit.

The previous module of Backstory could just match the current user utterance to the pre-set questions about the story of Magnus. It could not maintain context and hence could not have a conversation about life of Magnus. Converting this module to an FST made this possible. Now, Magnus can not only talk where it comes from but can also talk about its cousins and other family members.

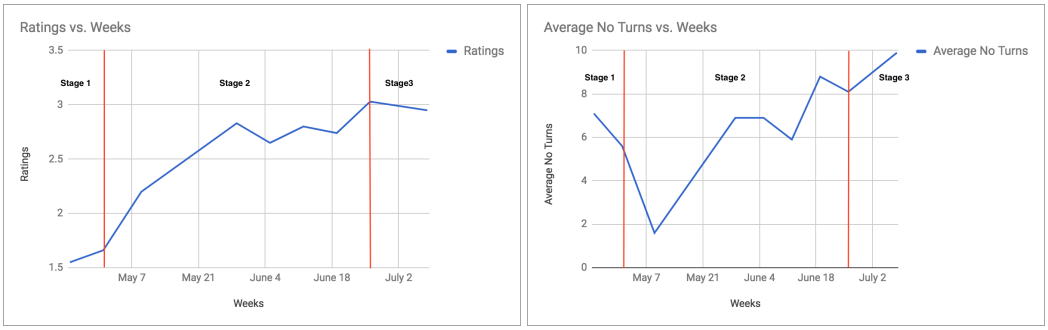
4.2 Issues

Inter-FST transitions. Exiting a FST at the right time and transitioning to the correct FST remains a challenge. There are two tasks involved here:

- Identify that the user is either frustrated or explicitly conveys that he/she wants to exit the current topic.
- Identify correctly the topic that the user wants to talk about and transition to the correct FST.

Model	Average for All Bots	Magnus
Ratings	2.92	3
Turn Error Rate	25.11%	23.54%
Avg # Turns	11	12
Median #turns	7	8
Median duration (sec)	97	112

Table 3: Metrics for CMU Magnus in the semifinal phase of the AlexaPrize



(a) Ratings across Stage 1, 2 and 3

(b) Average no. of dialog turns across Stage 1, 2 and 3

Figure 3: Ratings and Average number of dialog turns across time

To address the first issue, we do maintain a dictionary of utterances that denote that the user wants to exit the current topic. The dictionary is created by manually inspecting the previous conversation logs and identifying the utterances the users use to convey the frustration or change of topic.

Sequence of the FSTs. The sequence of FSTs was initially fixed. We change the order of the FSTs every 4 days to see how the order correlates to the scores we receive. We seek to find the best order of the FSTs.

5 System Results

The CMU Magnus system was interacting with users for the majority of the Alexa Prize competition. Our primary metrics for evaluating our performance were user ratings and dialog turns. Our goal was to maximize our average rating while also maximizing the number of dialog turns.

Figure 3a shows the variation of average user ratings across the three stages of development of CMU Magnus. As we can see, at the end of Stage 1, CMU Magnus had an average rating of 1.67, at the end of Stage 2 it was 2.74 and at the deployment of Stage 3, it was 3.03. Hence, we see improvement in ratings across the stages and this further strengthens our belief in the strategies used in each of the stages. Since, we have optimized for the length of the conversation, we see the improvement in the average number of dialog turns in Figure 3b. As of Aug 14 2017, we have achieved an average number of dialog turns of **12.1** which is better than systems having user ratings better than CMU Magnus. The two Figures 3 show that even though we can have long conversations with users, the users don't enjoy the conversations as they would with humans.

We believe that future research can improve both of these metrics and while this iteration of the competition has come to a close we believe that we can take these lessons learned into the next iteration. While Magnus did perform above average in the Alexa Prize competition (Table 3) our system can still clearly be improved.

6 Conclusion

It is hard to build a spoken dialog system without conversations from real users. With this opportunity of having real users interact with our system, we had many lessons to learn about the various strategies that could be adopted with different users in various contexts. In retrospect, we found that the need to

develop a stable testing system that could easily be installed and transferred to multiple machines, was of utmost importance. Additionally, we realized that detecting abuse is a complex and integral aspect of a live conversational agent. Lastly, automatic, contextualized conversations are still an impressively hard problem.

7 Acknowledgements

We are grateful for the guidance of Prof. Carolyn Rosé for her comments and criticism that greatly improved our system design. We thank our colleagues Aakanksha Naik, Sreecharan Sankaranarayanan, Eti Rastogi and Aditya Chandrashekhar for assistance in building some modules of the system.

References

[1] Yahoo Webscope. (2007). L6 - Yahoo! Answers Comprehensive Questions and Answers version (1.0). `https://webscope.sandbox.yahoo.com/catalog.php`, 2017. [Online; accessed 15-August-2017].

[2] P. Brown and S. C. Levinson. *Politeness: Some universals in language usage*, volume 4. Cambridge university press, 1987.

[3] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

[4] Y. Chen, Y. Zhou, S. Zhu, and H. Xu. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80, Sept 2012.

[5] C. J. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[6] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[7] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[8] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee, 2016.

[9] A. Ravichander, T. Manzini, M. Grabmair, G. Neubig, J. Francis, and E. Nyberg. How would you say it? eliciting lexically diverse data for supervised semantic parsing. In *The 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, Saarbrucken, Germany, August 2017.

[10] S. O. Sood, J. Antin, and E. F. Churchill. Using crowdsourcing to improve profanity detection. In *AAAI Spring Symposium: Wisdom of the Crowd*, volume 12, page 06, 2012.

[11] M. Tan, C. d. Santos, B. Xiang, and B. Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[12] J. Tiedemann. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248, 2009.

[13] D. Wang and E. Nyberg. A long short-term memory model for answer sentence selection in question answering. In *ACL (2)*, pages 707–712, 2015.