Reducing Friction between Users and Conversational AI using Markov Chains

Nik Bailey

Introduction

- What is a conversational AI?

- What challenges come from conversational AI?

Introduction

- What is a conversational AI?

- What challenges come from conversational AI?
    - Play Imagine Dragons - play maj and dragons

Introduction

- What is a conversational AI?

- What challenges come from conversational AI?
  - Play Imagine Dragons - play maj and dragons
  - Play Shallow by Lady Gaga - play shadow by lady gaga

Introduction

- What is a conversational AI?

- What challenges come from conversational AI?
  - Play Imagine Dragons - play maj and dragons
  - Play Shallow by Lady Gaga - play shadow by lady gaga
  - Don't ever play that song again - formal command is "Thumbs down this song"

Introduction

- What is a conversational AI?

- What challenges come from conversational AI?
  - Play Imagine Dragons - play maj and dragons
  - Play Shallow by Lady Gaga - play shadow by lady gaga
  - Don’t ever play that song again - formal command is “Thumbs down this song”
  - Turn the volume to half - formal command is “Volume five”

Problem: Friction

- Common way to reduce friction is to manually fix cases using rules and finite state transducers (FST)
  - Not scalable
  - Prone to error
  - Defective over time

- Another way is to identify frictions and ask annotaters to come up with correct form of query and then update the Alexa system
  - Not scalable
  - Expensive & time consuming

- Objective of research

Outline

- **Background Information**
  - Friction & Friction rate
  - Markov Chains
  - AI system overview
- Building the Markov Chain
  - Markov Chain in AI system
  - Dataset
  - Parts of the Markov Chain
- Experimentation & Results
  - Offline Analysis
  - Online Analysis

Background Information

- Friction & Friction Rate
- Markov Chains
- Conversational AI System overview

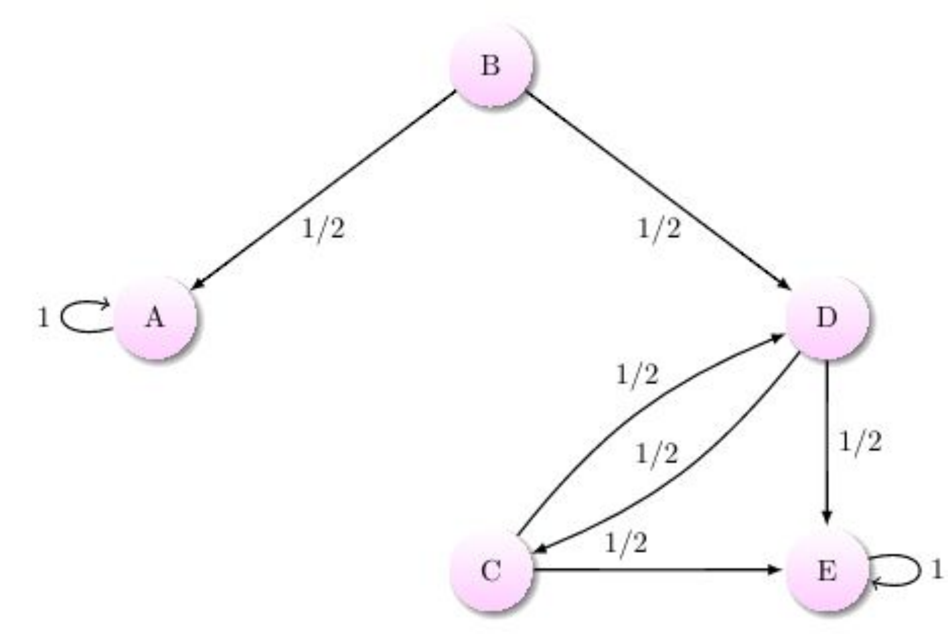
Friction & Friction Rate

- Friction - User dissatisfaction from AI response

- Friction Rate- Likelihood of causing friction between user and the AI
    - Computed by aggregating across utterances
    - Result of pre-trained neural model that leverages a user's utterance, the corresponding Alexa's response, and contextual signals to detect friction for every user-Alexa interaction exchange

Markov Chains

- Transient States
- Absorbing States
- Edges between states



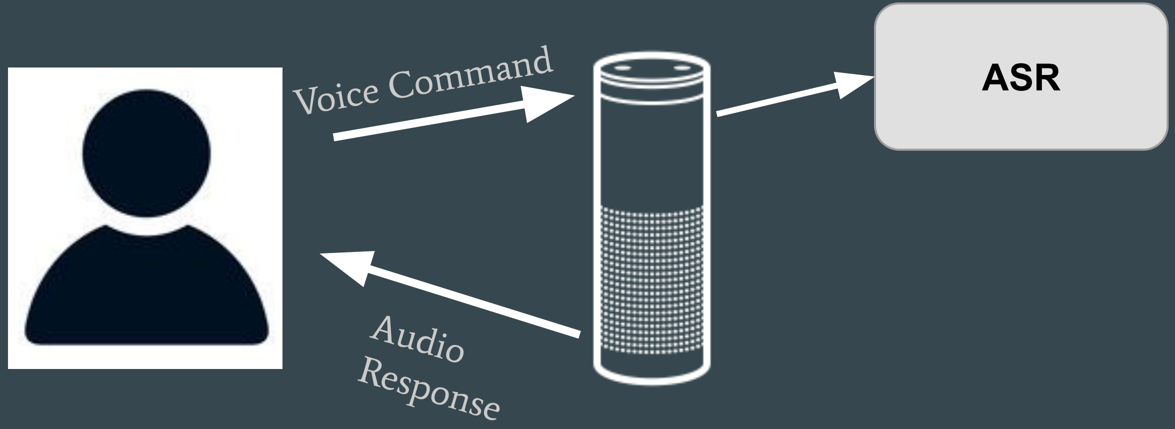
An absorbing Markov chain

AI System Overview

- Generally an Alexa system consists of:
    - Automatic Speech Recognition (ASR)
    - Natural Language Understanding (NLU)
    - Dialogue Management (DM)
    - Natural Language Generator (NLG)
    - Text to Speech (TTS)

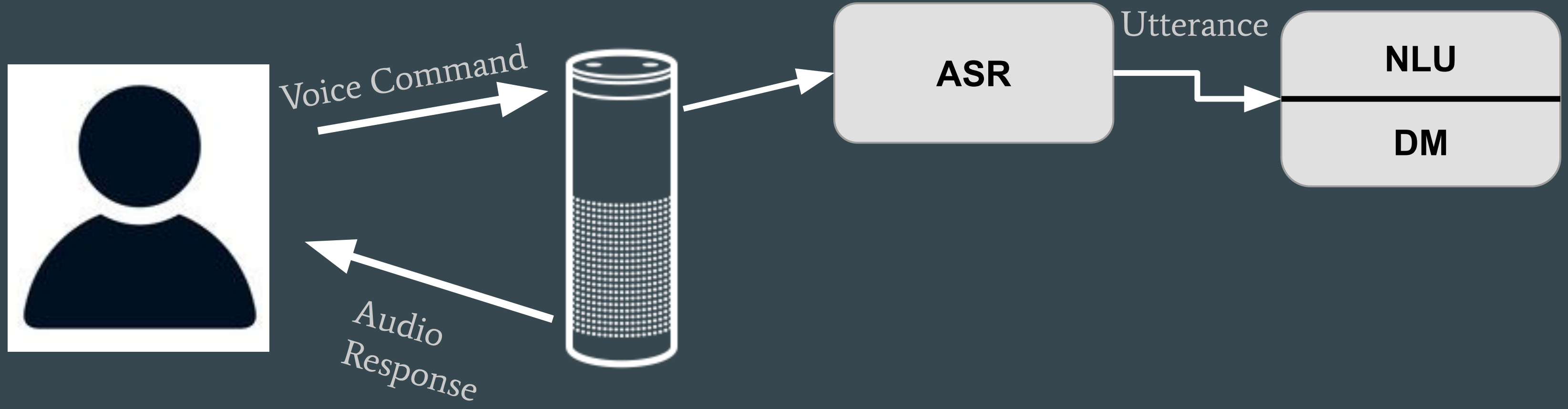
AI System Overview

- Generally an Alexa system consists of:
    - Automatic Speech Recognition (ASR)
    - Natural Language Understanding (NLU)
    - Dialogue Management (DM)
    - Natural Language Generator (NLG)
    - Text to Speech (TTS)



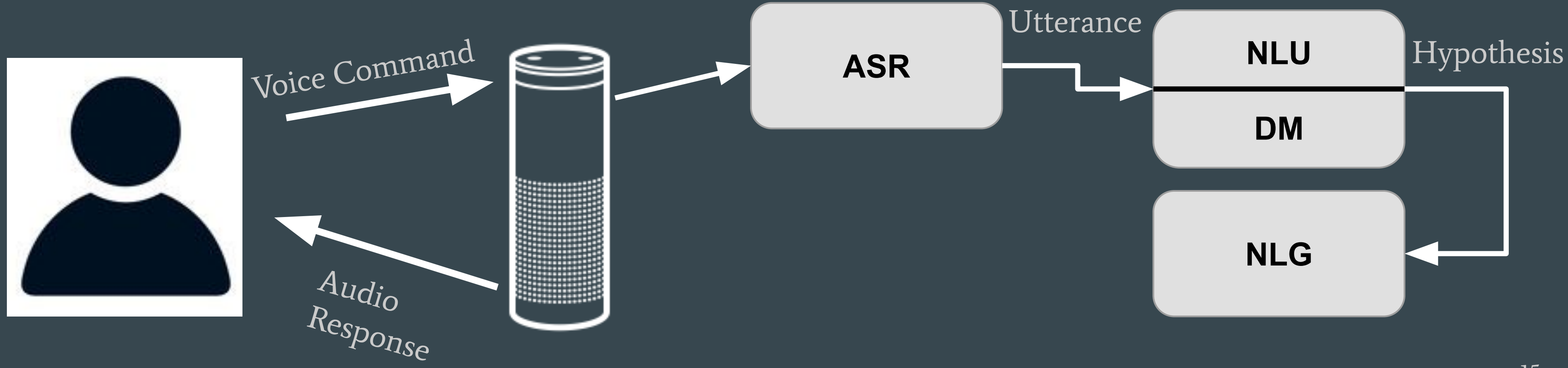
AI System Overview

- Generally an Alexa system consists of:
    - Automatic Speech Recognition (ASR)
    - Natural Language Understanding (NLU)
    - Dialogue Management (DM)
    - Natural Language Generator (NLG)
    - Text to Speech (TTS)



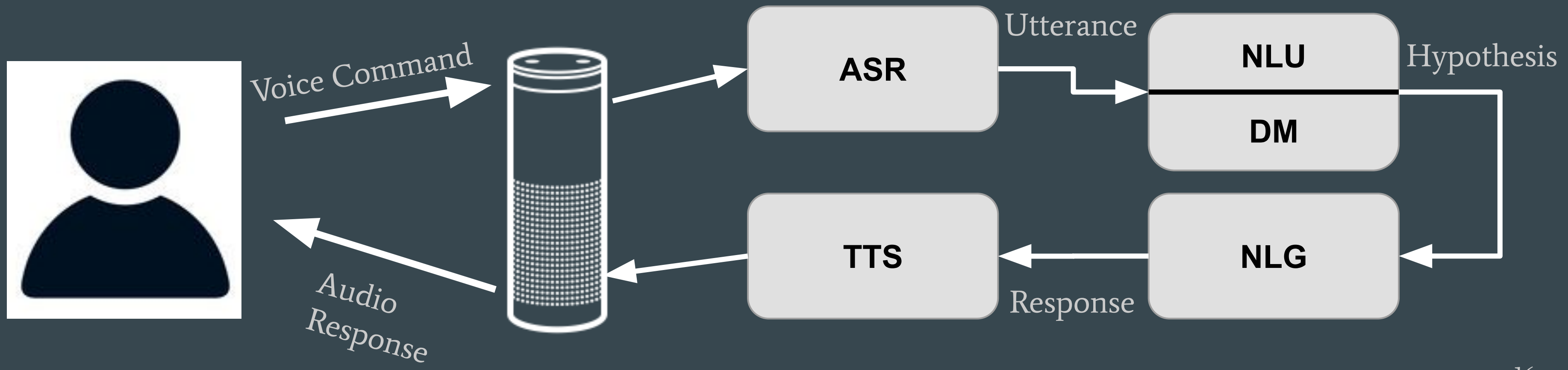
AI System Overview

- Generally an Alexa system consists of:
  - Automatic Speech Recognition (ASR)
  - Natural Language Understanding (NLU)
  - Dialogue Management (DM)
  - Natural Language Generator (NLG)
  - Text to Speech (TTS)



AI System Overview

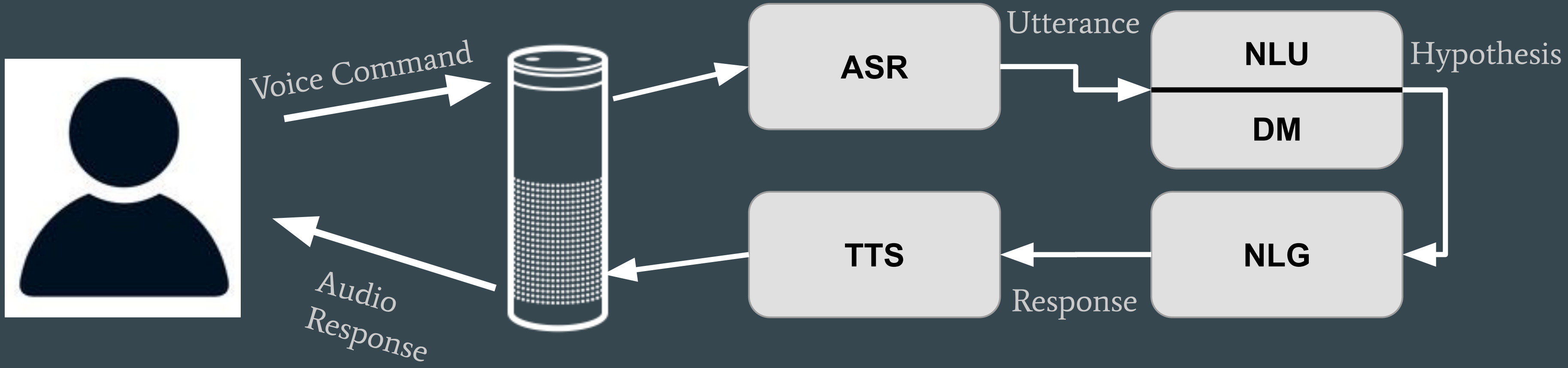
- Generally an Alexa system consists of:
    - Automatic Speech Recognition (ASR)
    - Natural Language Understanding (NLU)
    - Dialogue Management (DM)
    - Natural Language Generator (NLG)
    - Text to Speech (TTS)



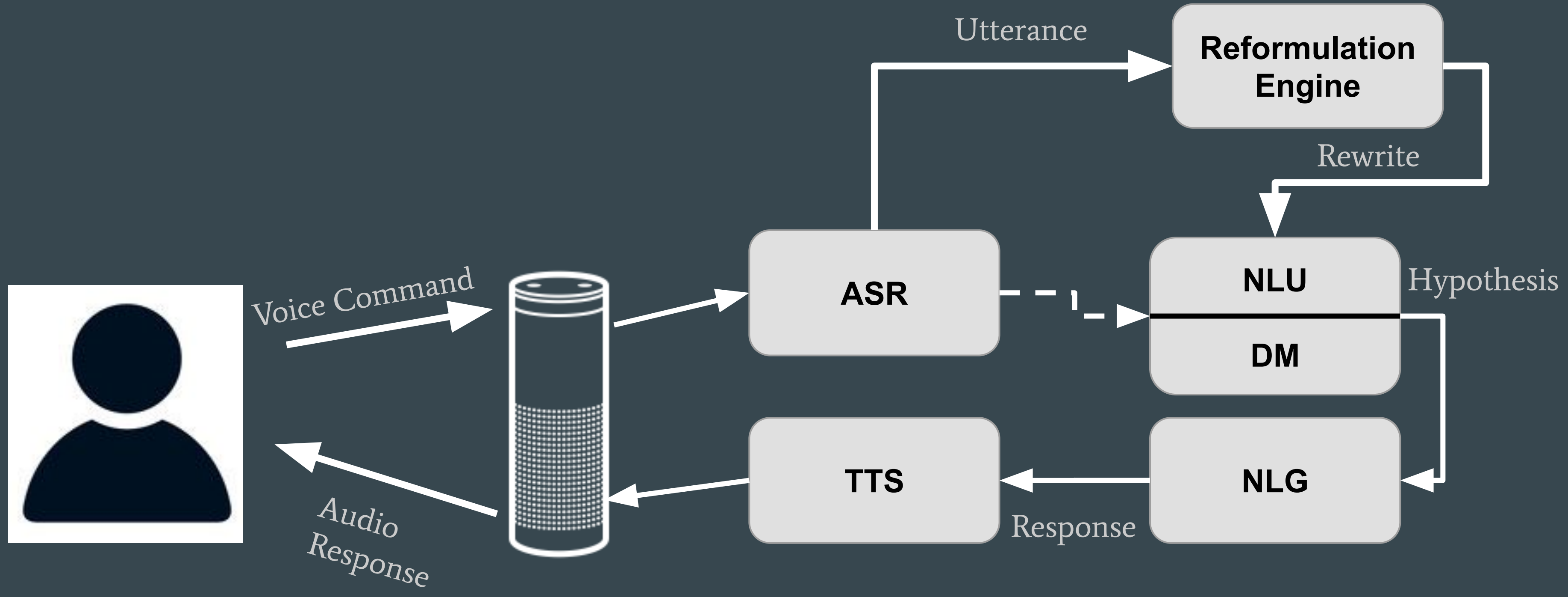
Outline

- Background Information
  - Friction & Friction rate
  - Markov Chains
  - AI system overview
- **Building the Markov Chain**
  - Markov Chain in AI system
  - Dataset
  - Markov Chain Attributes
- Experimentation & Results
  - Offline Analysis
  - Online Analysis

Markov Chain in the AI System

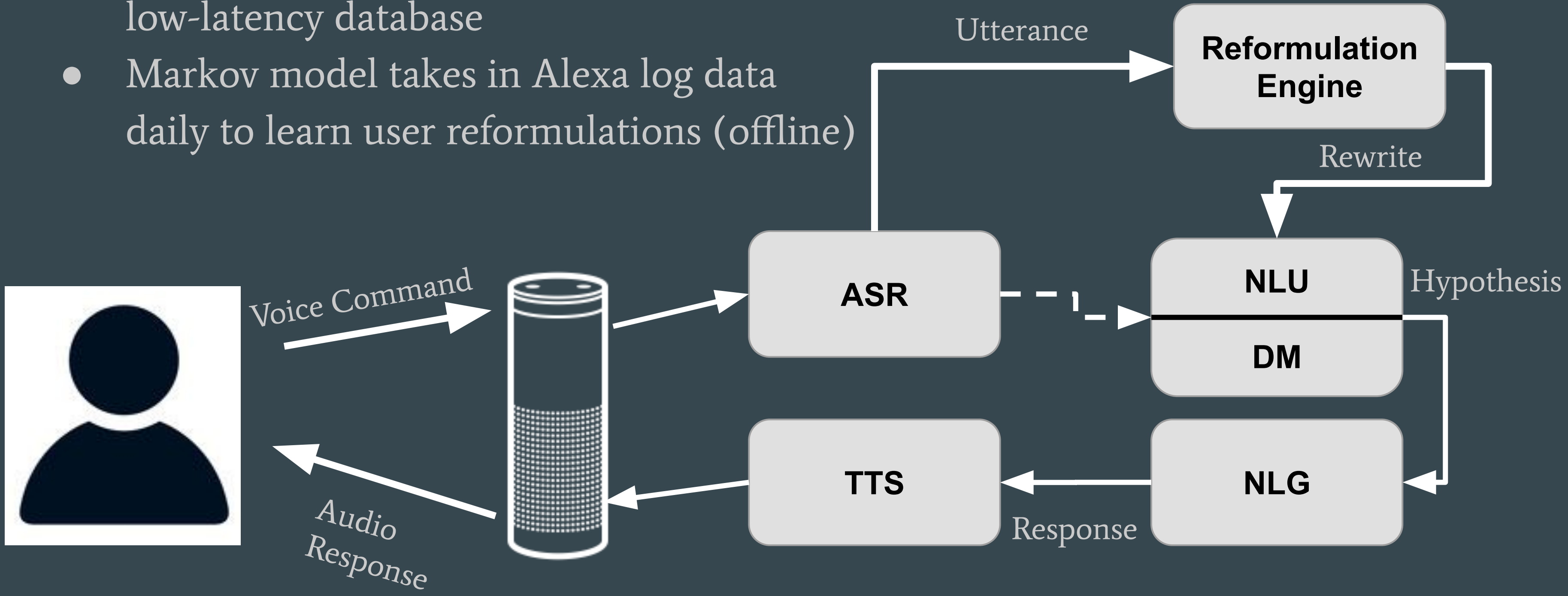


Markov Chain in the AI System

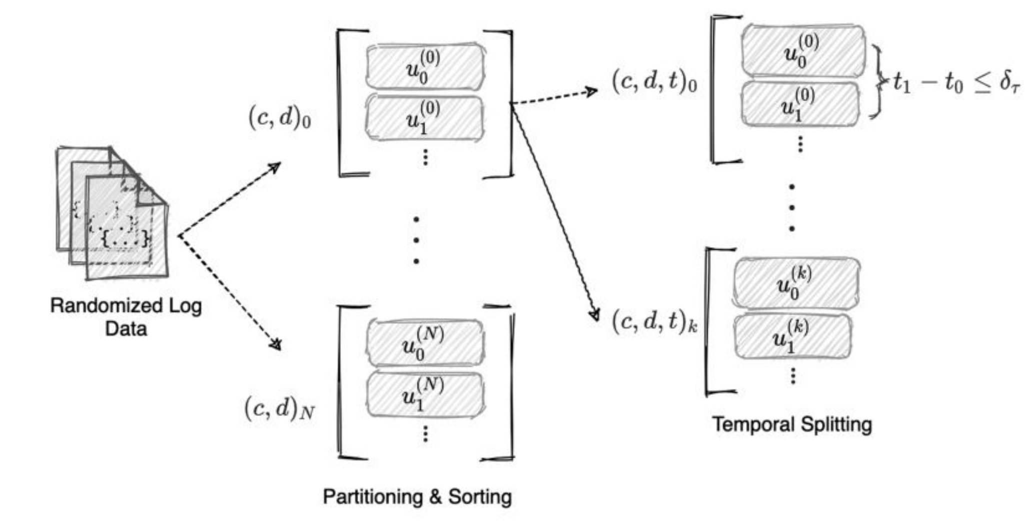


Markov Chain in the AI System

- Encapsulates access to high-performance low-latency database
- Markov model takes in Alexa log data daily to learn user reformulations (offline)



Dataset



- 3 month time period
- Millions of random users
- Sorted by user-device pair, then by individual conversations
  - Each individual session is a conversation between a user and their Alexa
  - Each session represented as a successive chain of utterances
  - Conversations separated by time delay > 45 sec

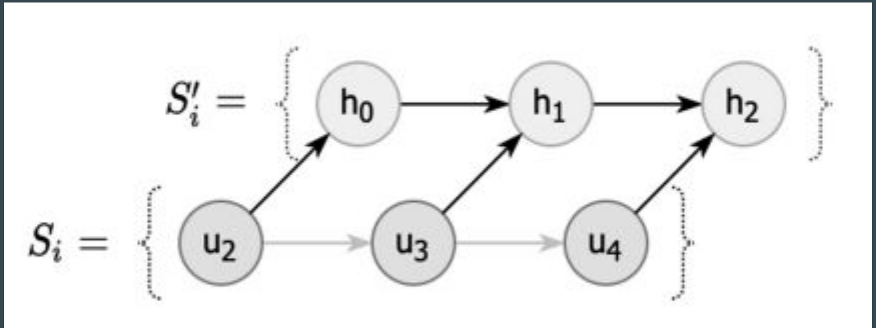
Markov Chain Attributes

- Utterance & Interpretation Space
- Transient States
- Absorbing States
- Creating highest probability of success

Utterance & Interpretation Space

- Utterance space U has sparse connection
  - Hard to connect similar sessions together
  - High degree of semantic and structural variance makes generalization difficult

- Uses the domain and intent classifier from the NLU
  - Encapsulates a latent distribution over the utterance space
  - Projected into interpretation space



Transient & Absorbing States

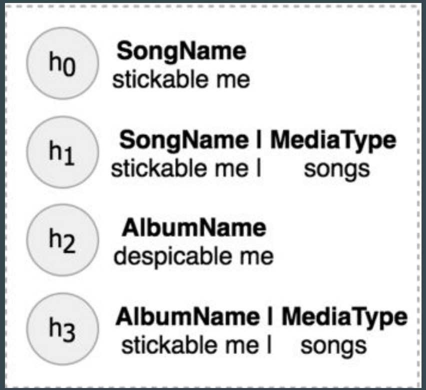
- Probability of moving from transient state *x* to transient state *y* is shown by

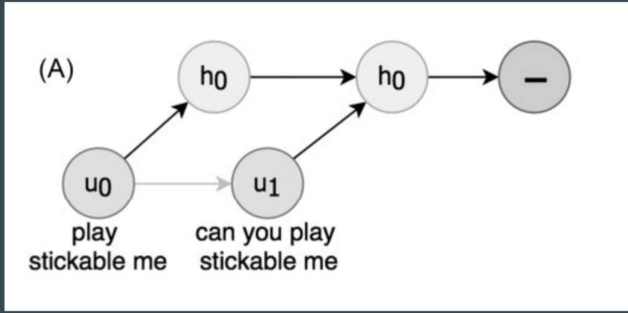
$$\frac{total\ number\ of\ times\ y\ is\ directly\ linked\ to\ x}{total\ occurrence\ of\ x\ aggregated\ across\ all\ sessions}$$

- Absorbing states are either a success (+) or a failure (-)
  - Use explicit and implicit feedback to determine absorbing states
    - Interjecting, canceling, or not responding to a request for clarification
  - Any feedback implies failure
  - Success is the absence of failure

Example

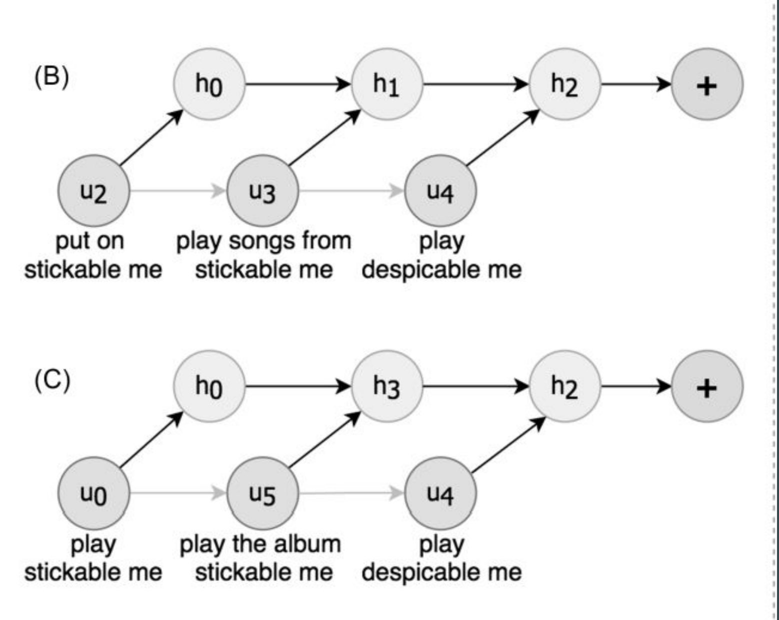
“Play Despicable Me”

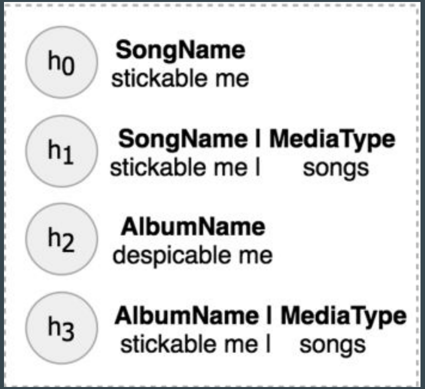




Example

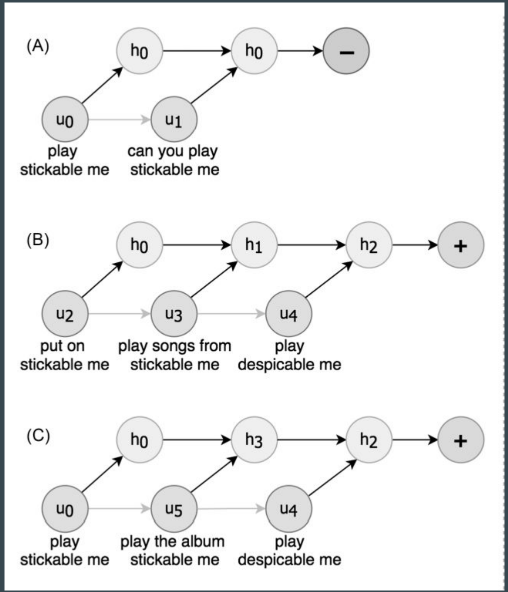
"Play Despicable Me"

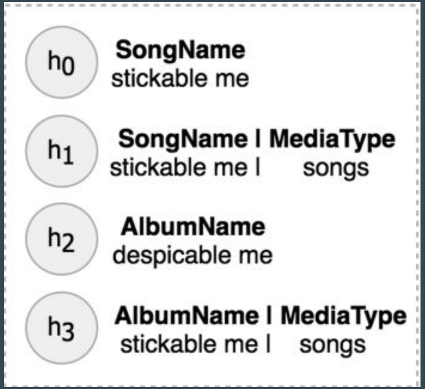


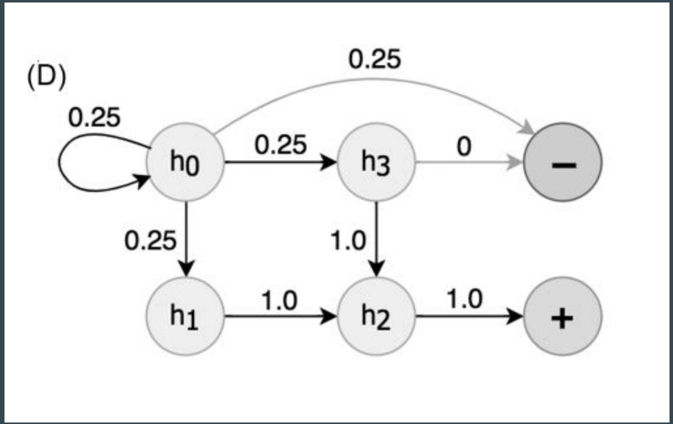


Example

“Play Despicable Me”





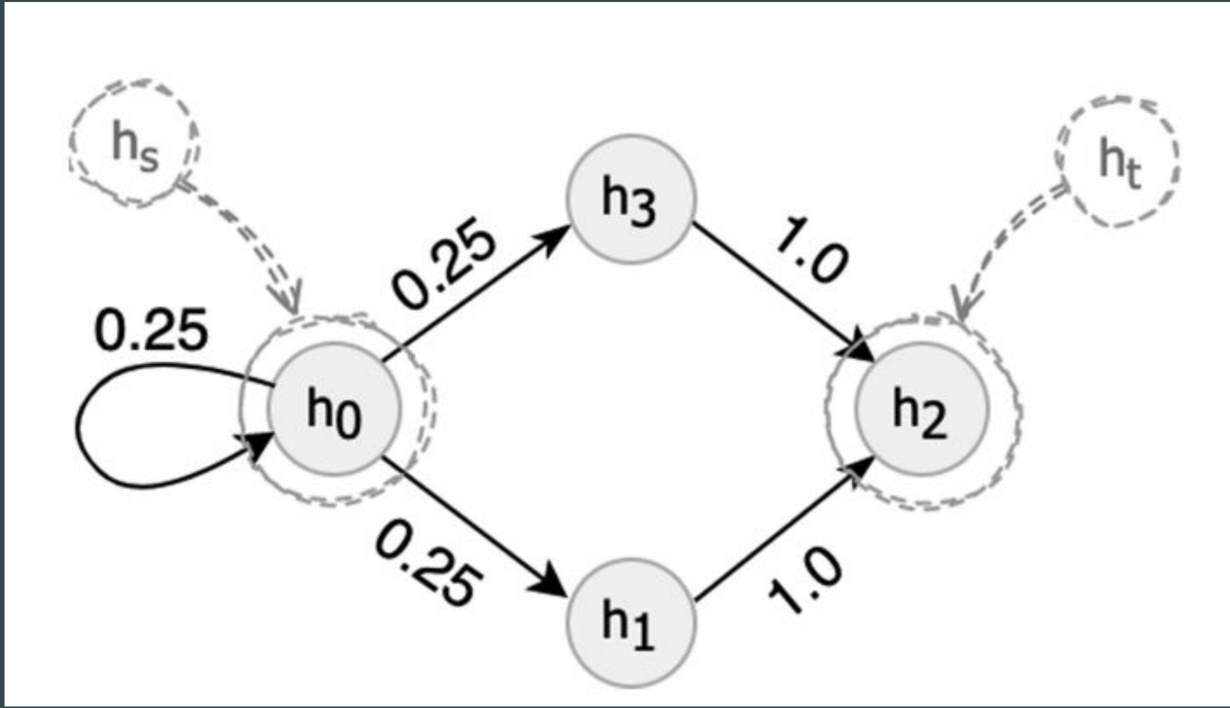


Markov Model - Highest Probability of Success

- Want to create Markov chain with highest chance of success
  - Given a transient state h_x, is there a transient state with that's more likely to succeed?

- Given an h_s and an h_t, determine what h^*_t is
  - If $h_s = h^*_t$ or $h_s \neq h^*_t$

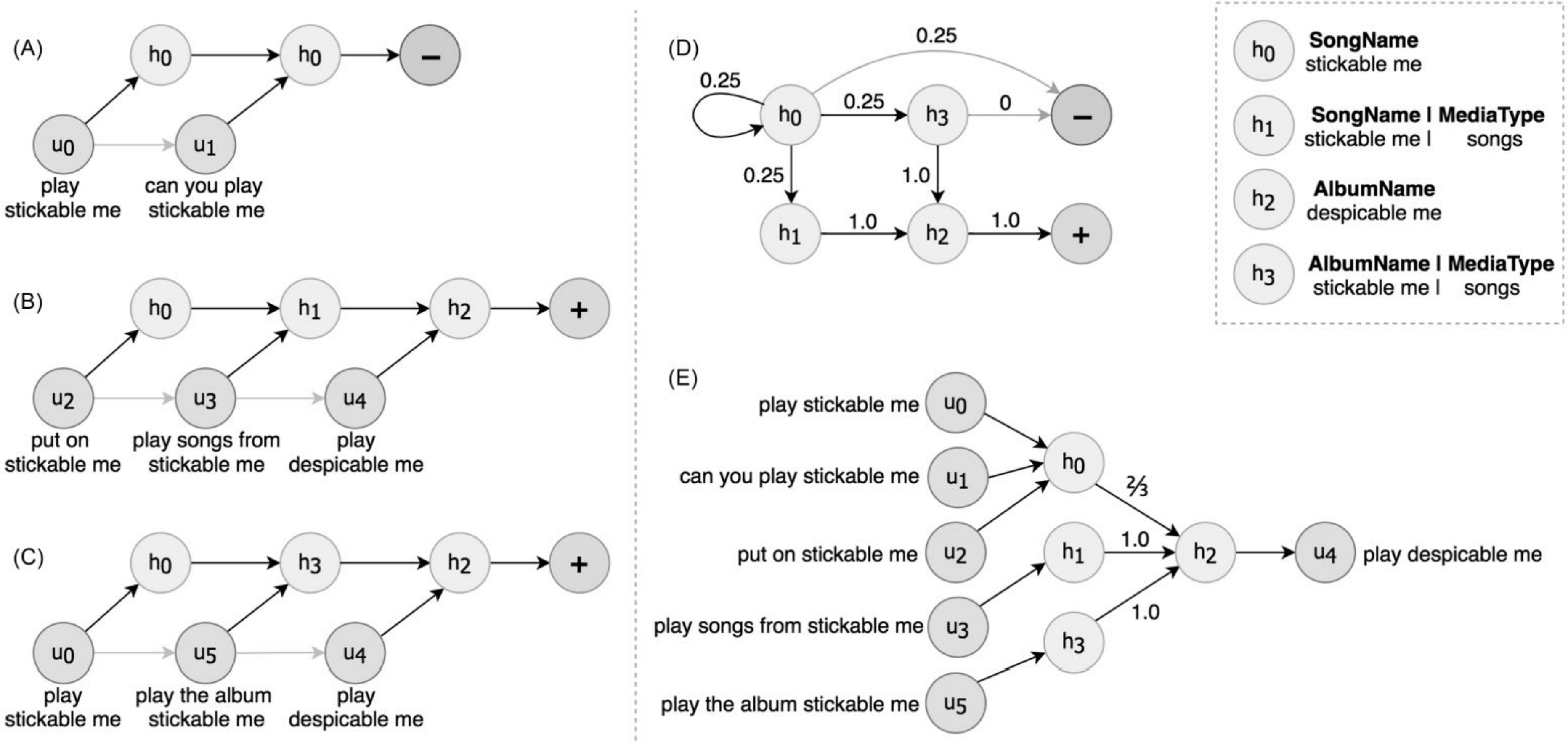
Markov Model - Highest Probability of Success



Markov Model - Highest Probability of Success

- In this example, $h_s \neq h^*_t$
  - h_s probability: 75%
  - h_t probability: 100%

- Model shows there's a reachable interpretation that when reformulated from h_s has a better chance of success than when not doing so


(A)
play stickable me
can you play stickable me
(B)
put on stickable me
play songs from stickable me
play despicable me
(C)
play stickable me
play the album stickable me
play despicable me
(D)
0.25
0.25
0.25
0
0.25
1.0
1.0
1.0
SongName
stickable me
SongName | MediaType
stickable me | songs
AlbumName
despicable me
AlbumName | MediaType
stickable me | songs
(E)
play stickable me
can you play stickable me
put on stickable me
play songs from stickable me
play the album stickable me
2/3
1.0
1.0
play despicable me

Markov Model - Highest Probability of Success

- Similar idea can be used on the utterance space
- Reformulate to get a u^*_t that is more successful than u_s
- Self-partitioning allows the model to only target utterances that are likely to be defect
    - Drives self-learning nature without needing human interaction

Outline

- Background Information
  - Friction & Friction rate
  - Markov Chains
  - AI system overview
- Building the Markov Chain
  - Markov Chain in AI system
  - Dataset
  - Markov Chain Attributes
- **Experimentation & Results**
  - Offline Analysis
  - Online Analysis

Offline Analysis - Creating Baseline Model

- Sequence-to-sequence architectures have been the foundation for many neural machine translation and sequence learning tasks

- Used long short-term memory based (LSTM) model to produce rewrites
  - Pointer-generator model
  - Trained with 3 months of mined rephrase data

- Used to rewrite same utterance Markov chain rewrites

Offline Analysis

- Annotated 5679 unique utterance-rewrite pairs generated by Markov model

<u>Markov Chain Model</u>

- 93.4% accuracy
- win/loss ratio 12.0

<u>Pointer-Generator model</u>

- 55.2% accuracy

Offline Analysis - Markov Chain Benefits

- Aggregates all 3 months of data
- Accounts the frequency of transitions, whereas the pointer-generator model only has unique rephrase pairs
- Uses interpretation space to further compact and aggregate utterances
- Capable of identifying if an utterance is successful, and so won’t rewrite

Offline Analysis - Markov Chain Failures

- Rewrite is generic
  - “Play”
  - “Shuffle my songs”
  - Usually happens when first utterance isn’t successful, following utterance lose information
- Rewrite changes intention by changing song name or artist name
  - Period of time where original utterance wasn’t successful, so user asked for similar song

No.	Original utterance	Rewrite	Label
1	Play maj and dragons	Play imagine dragons	Good
2	Play shadow by lady gaga	Play shallow by lady gaga	Good
3	Play rumer	Play rumor by lee brice	Good
4	Play sirius x. m. chill	Play channel fifty three on sirius x. m.	Good
5	Play a. b. c.	Play the alphabet song	Good
6	Don’t ever play that song again	Thumbs down this song	Good
7	Turn the volume to half	Volume five	Good
8	Play island ninety point five	Play island ninety eight point five	Good
9	Play swaggy playlist	Shuffle my songs	Bad
10	Play carter five by lil wayne	Play carter four by lil wayne	Bad

Online Analysis

- A/B Testing setup - Rewrites against no rewrites

- Two week rewrite performance
  - 30% average defect rate reduction
  - Defect based on machine learning model that scores user dissatisfaction at every turn

- Nine week randomized control trial
  - As defects decreased, user engagement increased

- Win/loss ratio after 3 months - 11.8
- Monitored weekly for 15 months

Conclusion - Fulfilling Research Objectives

- Scalable
- Updates database regularly
- Time-efficient
- Self-Learning

Questions?

- https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/15102
- https://brilliant.org/wiki/absorbing-markov-chains/#:~:text=A%20simple%20example%20of%20an,they%20will%20stay%20there%20forever.