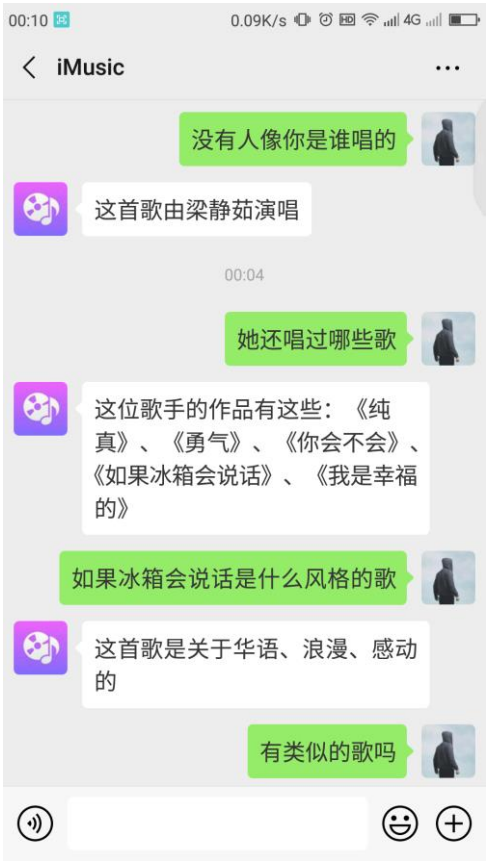
# Demo presentation

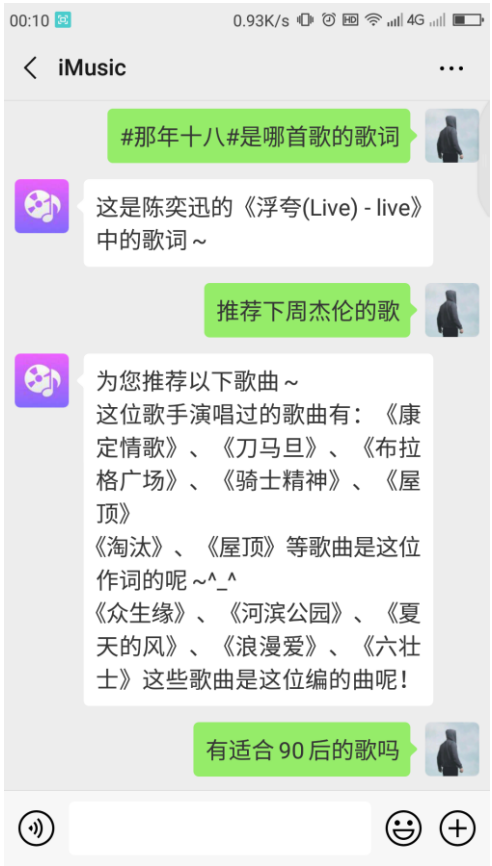
- Use the Wechat interface to provide the service, some screenshots:

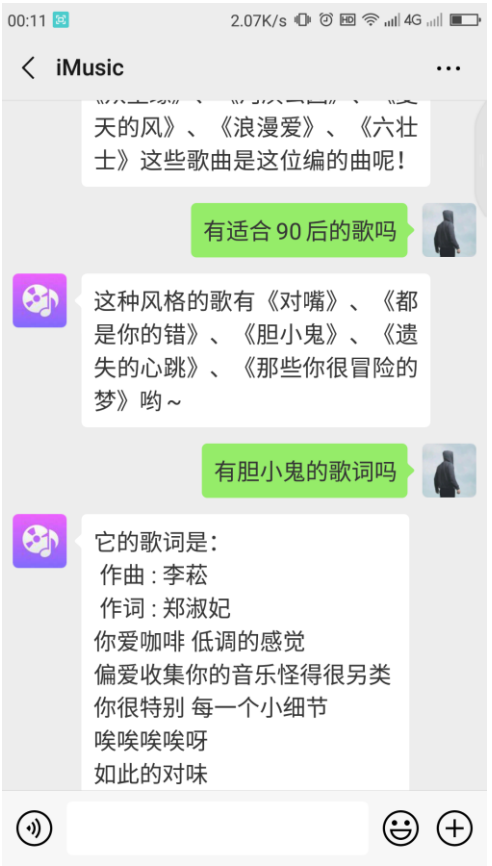




# Demo presentation

- Use the Wechat interface to provide the service, some screenshots:

# *A Dialog System in Music Domain*


Jia Chen
DCST, Tsinghua Univ.
cj18@mails.tsinghua.edu.cn

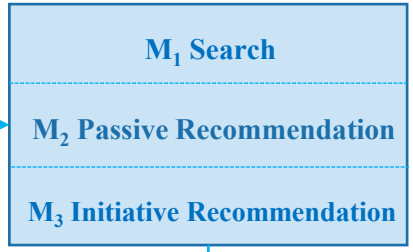
Leilan Zhang
DCST, Tsinghua Univ.
zll17@mails.tsinghua.edu.cn

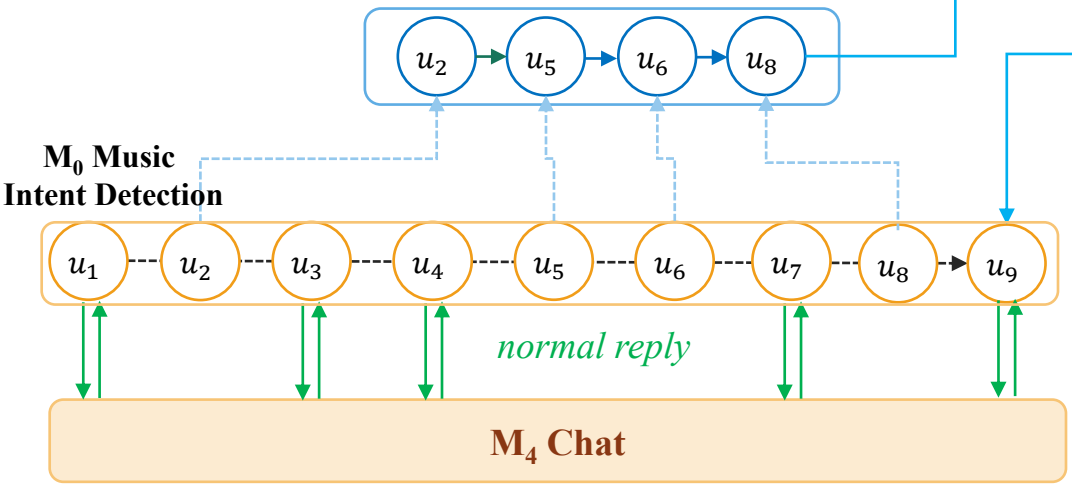

*05-13-2019*

# Introduction

- Framework
  - Context-aware recommendation
    - Search
    - Passive Recommendation
    - Initiative Recommendation
  - Chat module


**Context-aware recommendation**




**$M_1$** 用户具有清晰、明确的信息意图，例如搜索一首歌曲的演唱者、歌词等等；
**$M_2$** 用户具有一定的信息意图，例如推荐周杰伦的几首歌，推荐梁静茹治愈风格的歌曲等等；
**$M_3$** 用户当前没有体现任何意图，我们需要使用上下文数据进行推荐，例如用户：给我推荐几首歌吧；
**$M_4$** 没有任何音乐意图，使用chat模块进行回复，同时不添加到context记录中。

# Experiment Procedure

- Data collection
  - Dialogue data, collect, filter, pair-wise reply;
  - Music data, crawled from *Wangyiyun*, in JSON format;
- Model construction
  - Search module;
  - Passive recommendation;
  - Initiative recommendation;
  - Chat Module
- Evaluation

# Data Collection: *Music Data Analysis*

| # Song | 5,994 |
|---|---|
| # Artist | 2,676 |
| # Labels | 69 |
| # Avg. Comments | 11.485 |

Table: Statistics of Music Data

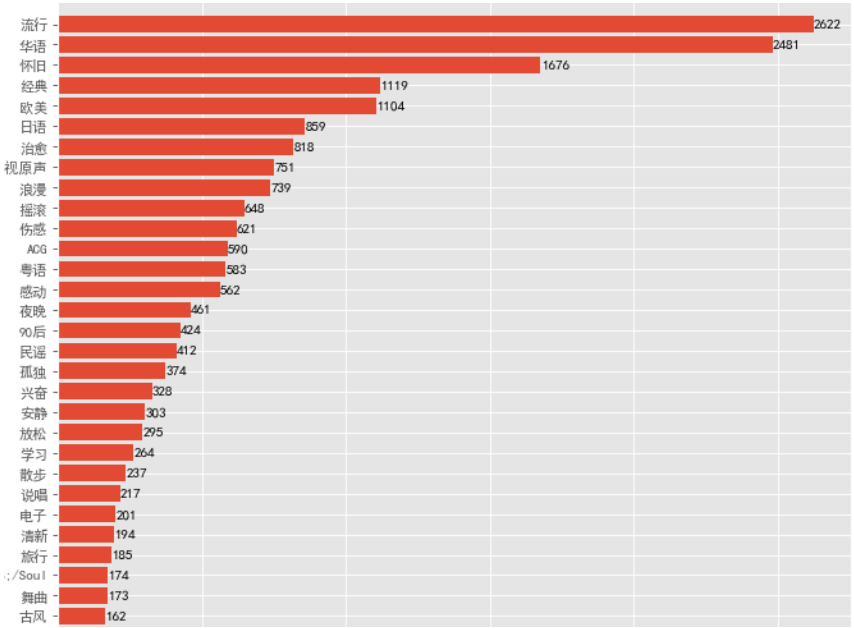



Figure: Top 30 labels

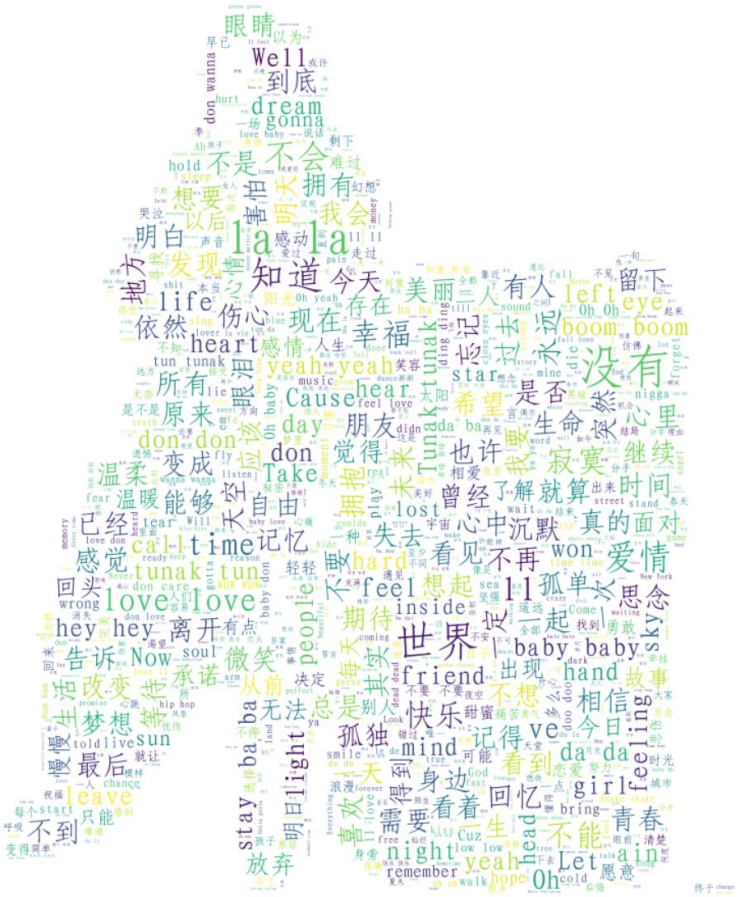
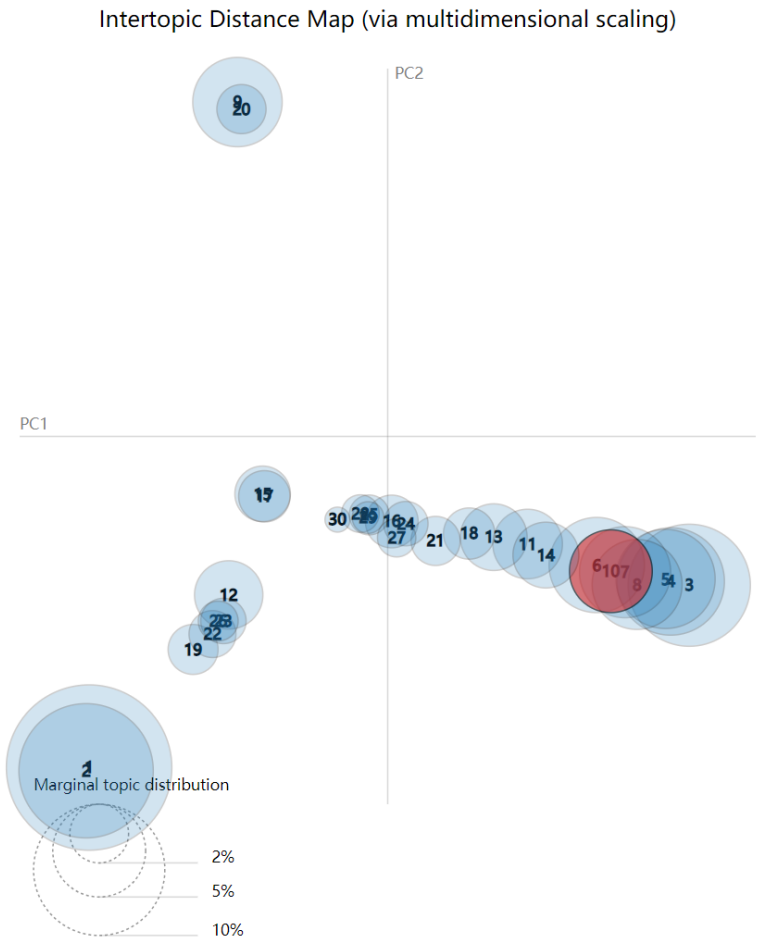
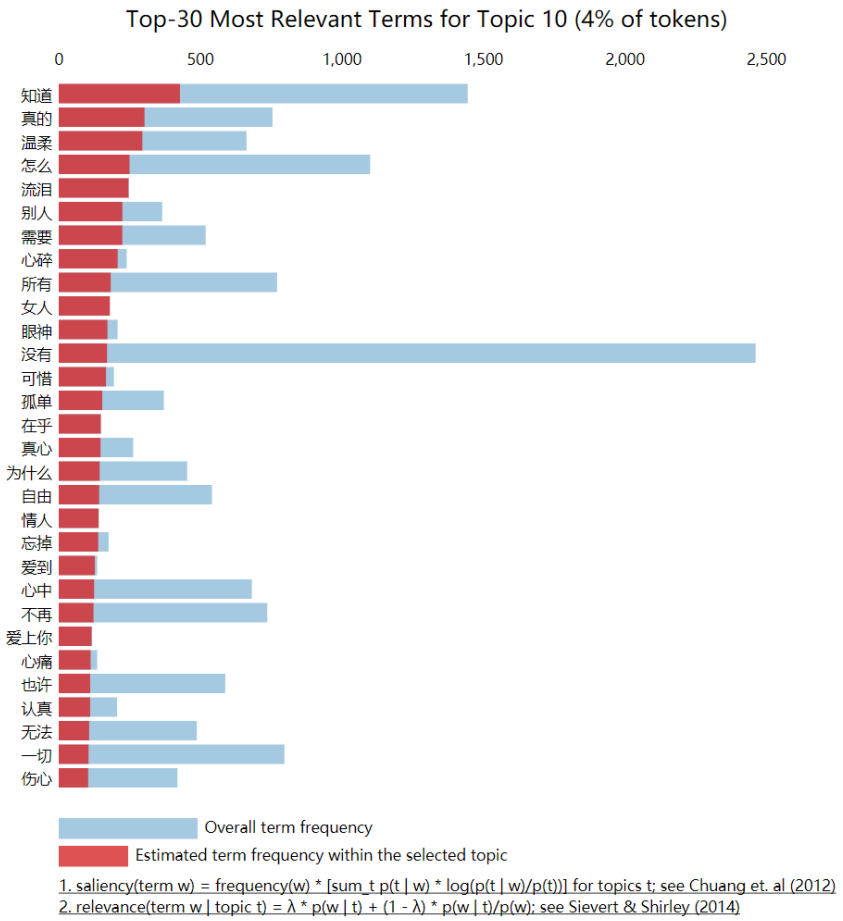



Figure: Lyric Cloud

# *LDA: Music Topic 10*



# $M_0$: Music Intent Detection

- Binary Classification
  - Positive corpus: we generate the corpus with music intent by using pre-defined patterns which will be used in the dialogue system.
  - exp. We defined following patterns for style or genre, and filled those brackets with tags or topics (like 怀旧, 摇滚, 民谣 etc.):

```
'想 听 一些 {} 的 歌','{} 的 歌词 是 什么','来 一首 {} 的',\
'找 一首 {} 的', '要 {} 的', '{} 的 歌 有 吗', '{} 的 有 吗', \
'有 {} 的 歌 吗', '有 {} 的 吗', '我 想 找 风格 是 {}', \
'我 想 找 标签 是 {}', '我 想 找 主题 是 {}', '那 首 {}',\
'来 一首 style 是 {}', '来 一首 情调 是 {}', '来 一首 画风 是 {}', \
'找 一首 风格 是 {}', '找 一首 标签 是 {}', '找 一首 主题 是 {}', \
'找 一首 style 是 {}',  '找 几首 主题 是 {}', '找 首 style 是 {}'
```

  - We also generated data for other slots using the same method: song names, singers and lyric writers were all filled into those brackets.

# $M_0$: Music Intent Detection

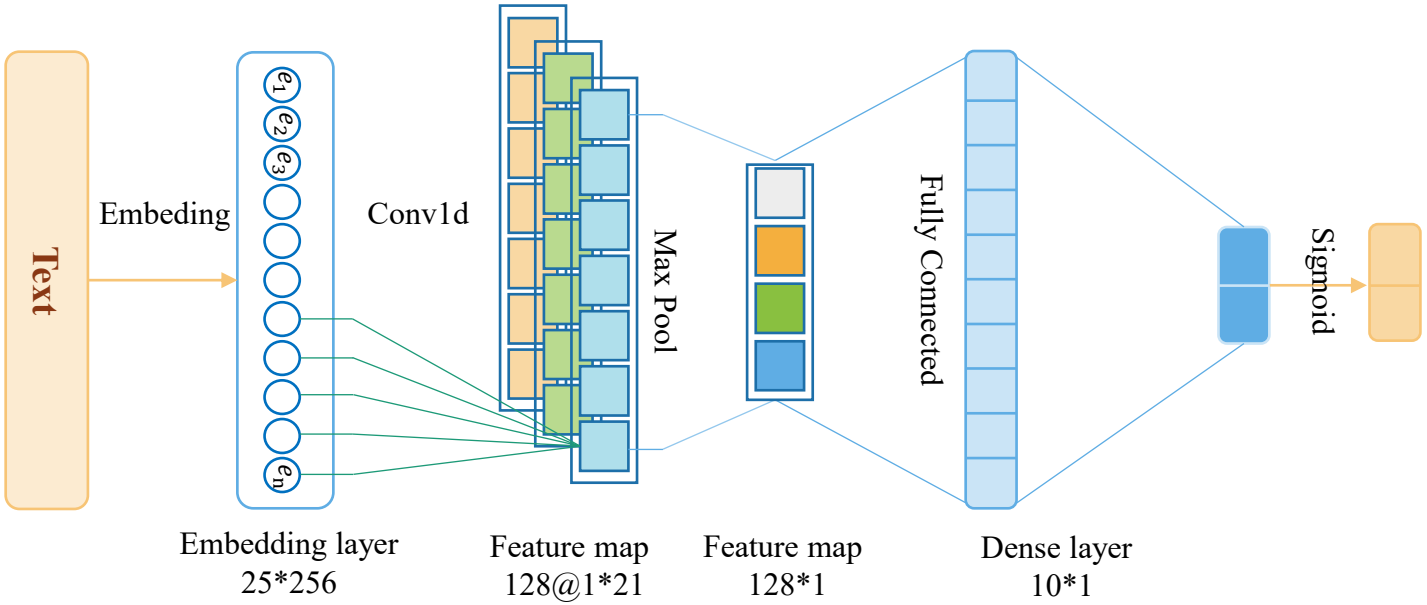
- Binary Classification
  - According to practical experience, some extra cases were also added into the train data, a piece of them are displayed as following:

```
'推荐一些歌吧','推荐点歌吧','有什么推荐的歌','最近有哪些热门歌',\
'近期的热门歌有哪些','还有什么类似的推荐吗','还有什么类似的歌吗',\
'是哪首歌的歌词呢','还有类似的歌吗','推荐一些歌吧','还有什么相似的歌',\
'推荐几首歌','推荐类似的歌','求推荐歌曲','推荐好听的歌','有什么好听的吗',\
'有哪些流行的歌','有哪些好听的歌','他还唱了什么歌','他还唱过哪些歌',\
'她还唱过哪些歌','她还唱过什么歌嘛','有什么故事呢','还给哪些作了词',\
'有什么好听的歌','求推荐好听的歌','有哪些好听的歌','我想听歌'
```

  - Negative corpus: the chat corpus we refine from JD dialogue data and the chat bot corpus, with no music intents.
  - Finally obtained 210394 positive cases and 232046 negative cases.

# $M_0$: Music Intent Detection

Use CNN to accomplish the classification




| | | |
|---|---|---|
| 最近 有 什么 好 听的歌 吗 | --> | 1 |
| 你 买 新车 了 | --> | 0 |
| 想 听 爱 的 主打歌 | --> | 1 |
| 我 好 无聊 啊 | --> | 0 |
| 舒缓 一点 的 歌 有吗 | --> | 1 |
| 有 张国荣 的 歌 吗 | --> | 1 |
| 想要 首 好听 的 | --> | 1 |
| 推荐 几步 电影 吧 | --> | 0 |
| 推荐 两 首 新歌 | --> | 1 |
| 楼下 新开 了 一家 小吃店 | --> | 0 |
| 梁静茹 的 治愈 风格 | --> | 1 |

| | prec | Recall | f1 |
|---|---|---|---|
| train set (309708) | 0.99 | 0.99 | 0.99 |
| test set (132732) | 0.97 | 0.96 | 0.96 |

# $M_1$: Search Module & $M_2$: Passive Recommendation

- Pre-defined search types and natural language patterns
  - Search for information of a song, e.g. the artist, writer, composer, style, lyric, comments and stories, etc.
  - Search for the song given a piece of lyric, return the song name and the artist
  - Search for songs of an artist, a writer, a composer or a tag
  - Composite search, such as songs of an artist given the style...

  → Totally manually designed **27** main search types and **100+** natural language patterns

- Passive recommendation
  - Based on search module, using some information like user **comments and likes** to recommend most popular songs that satisfy users' needs.

# $M_3$: Initiative Recommendation

- Utilize "wisdom of crowds" and the user's personalize data

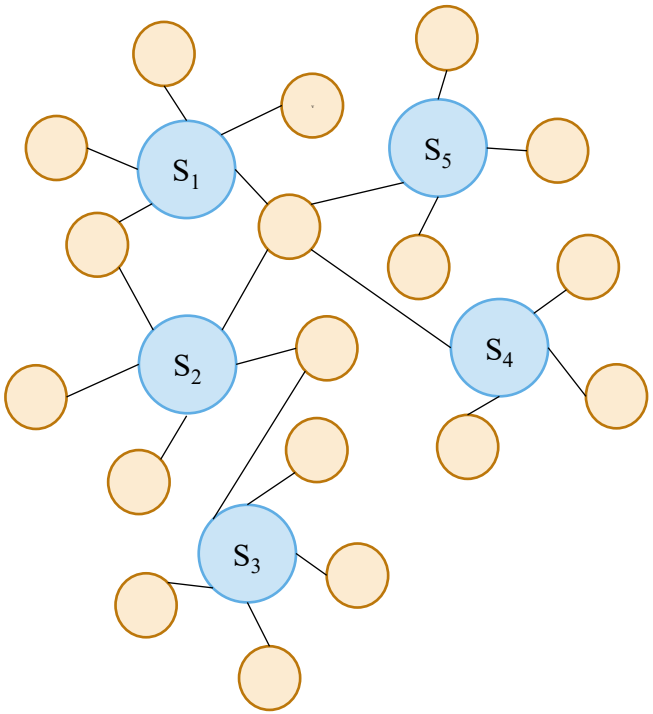



Figure: Song-user comment graph

**Algorithm**

1. Build the song-user graph;
2. Use node2vec to calculate the embedding of each song node and user node;
3. For a new user, we initialize her preference embedding by:

$$p_u^0 = \frac{1}{N}\sum_{i=1}^{N} E_{u_i}$$

4. We can collect user feedback for a song in the conversation, where we make a sentimental analysis on the user's response by:

$$f^t = sentiment(r^t), f^t \in [-1,1]$$

5. We update the user's preference embedding by:

$$p_u^t = p_u^{t-1} + \gamma f^t \sum E_{s_k^t}$$

6. At each time, if the system needs initiative recommendation, we find top $k$ most similar users and recommend the songs from their lists.

# $M_4$: Chat Module

- Framework

#1 Encoder(Bi-GRU)

$$\boldsymbol{h} = [\boldsymbol{h_1}, \boldsymbol{h_2}, \cdots, \boldsymbol{h_L}], \text{ where } \boldsymbol{h_i} = [\overrightarrow{\boldsymbol{h_i}}; \overleftarrow{\boldsymbol{h_i}}]$$
$$\overrightarrow{\boldsymbol{h_i}} = \text{RNN}(\overrightarrow{\boldsymbol{h_{i-1}}}, x_i) \quad \overleftarrow{\boldsymbol{h_i}} = \text{RNN}(\overleftarrow{\boldsymbol{h_{i+1}}}, x_i)$$

#2 Decoder(with Attention)

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

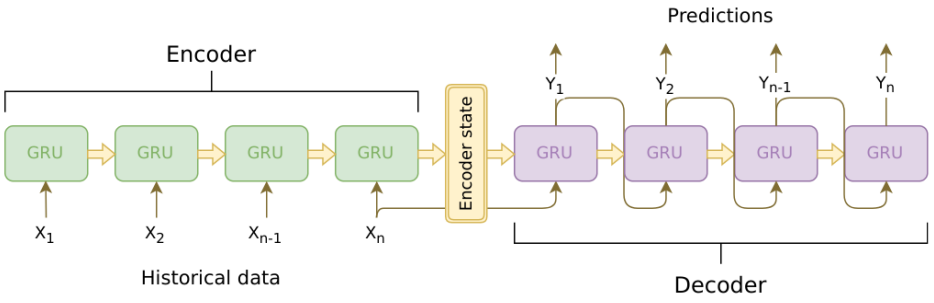



Figure: Seq2seq model framework

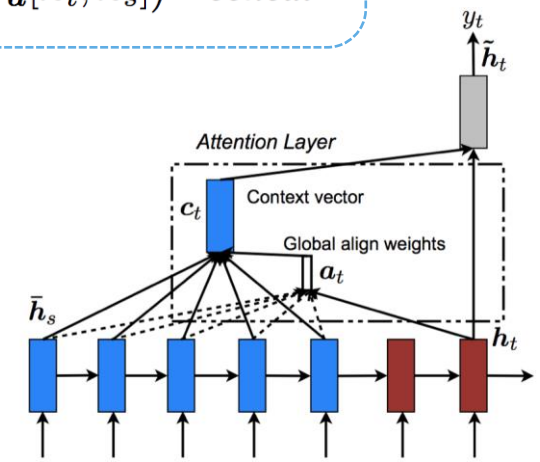



Figure: Attention Mechanism

# $M_4$ Chat Module

- Setups
  - *Hidden size = 256*
  - *Training Epoch = 200,000*
  - *Attention = general*
  - *Learning rate = 0.0001*
  - *Max length = 30*
  - *batch size = 64*
  - *dropout rate = 0.1*
- Evaluation
  - ***3-scale labeling( 0-bad, 1-good, 2-perfect)***
  - *0: OOV error, poor syntactic structure and poor semantic cohesion.*
  - *1: good syntactic structure or good semantic cohesion.*
  - *2: good syntactic structure and good semantic cohesion, perfect reply*

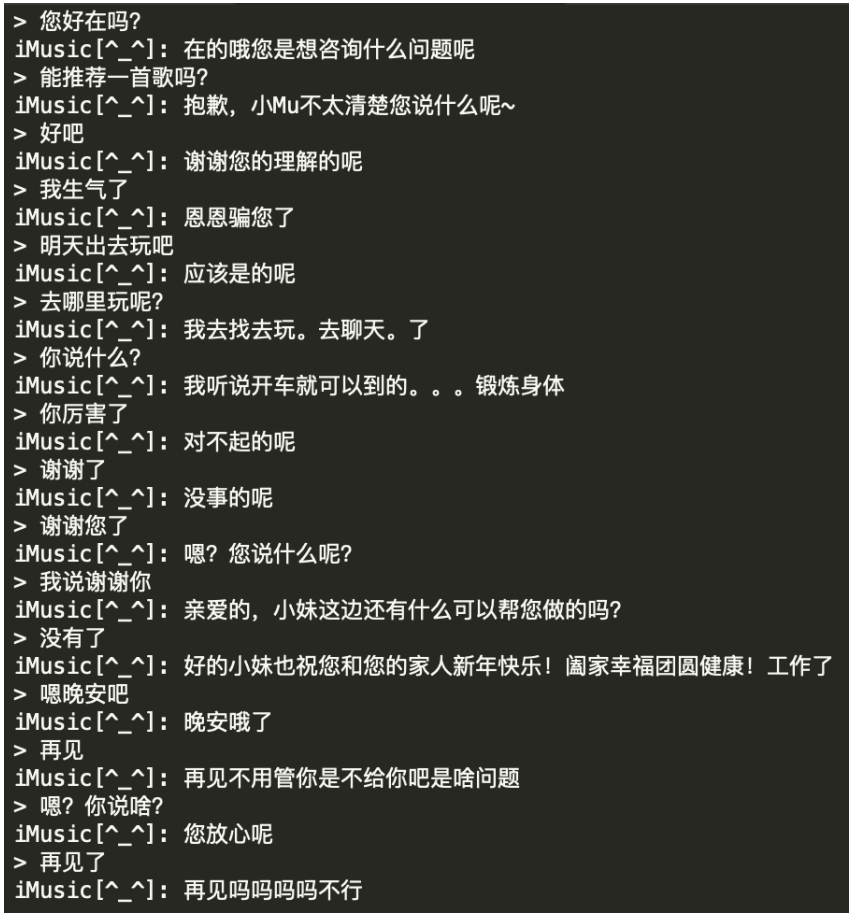



Figure: Sample dialogue

# $M_4$ Chat Module: Evaluation

| | *0(Bad)* | *1(Good)* | *2(Perfect)* | *Fleiss Kappa κ* |
|---|---|---|---|---|
| 10000 | 0.12 | 0.44 | 0.44 | **0.7261** |
| **20000** | **0.12** | **0.33** | **0.55** | 0.4745 |
| 30000 | 0.18 | 0.31 | 0.51 | 0.5301 |
| 40000 | 0.12 | 0.37 | 0.51 | 0.4223 |
| 50000 | 0.15 | 0.37 | 0.48 | 0.4002 |

Table: Training Sample Performance

- Performances on each step are close.
- Generally, the 20000 checkpoint has the best overall performance.
- Bad cases only account for a little proportion (10%-20%).

| | *0(Bad)* | *1(Good)* | *2(Perfect)* | *Fleiss Kappa κ* |
|---|---|---|---|---|
| 10000 | 0.58 | 0.27 | 0.15 | 0.4515 |
| 20000 | 0.65 | 0.24 | 0.11 | 0.3728 |
| 50000 | 0.65 | 0.25 | 0.1 | 0.4199 |
| 100000 | 0.68 | 0.24 | 0.08 | 0.4524 |
| **200000** | **0.48** | **0.29** | **0.23** | **0.6289** |

Table: External Sample Performance

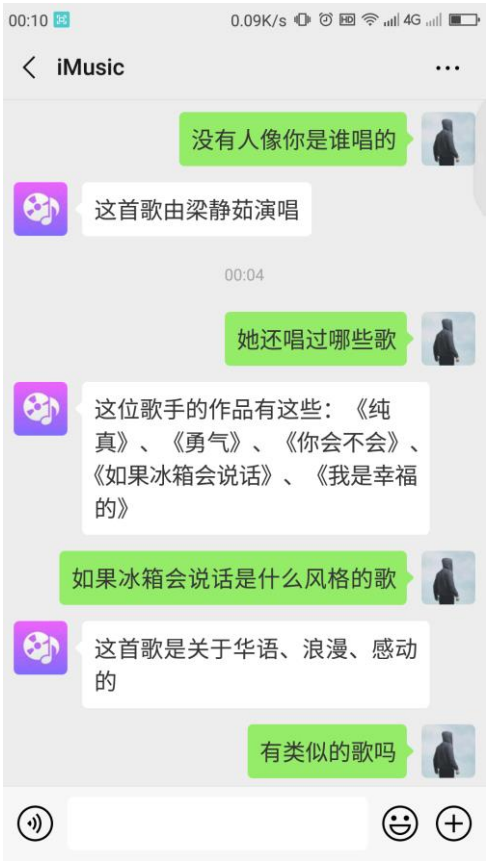
- Performances on each checkpoints are very different.
- Generally, the 200,000 checkpoint has the best overall performance.
- Bad cases account for a large proportion (about 50%).

**Fleiss Kappa: 0-0.2(slight), 0.21-0.40(fair),* ***0.41-0.6(moderate), 0.61-0.8(substantial),*** *0.8-1(almost perfect)*

# Demo presentation

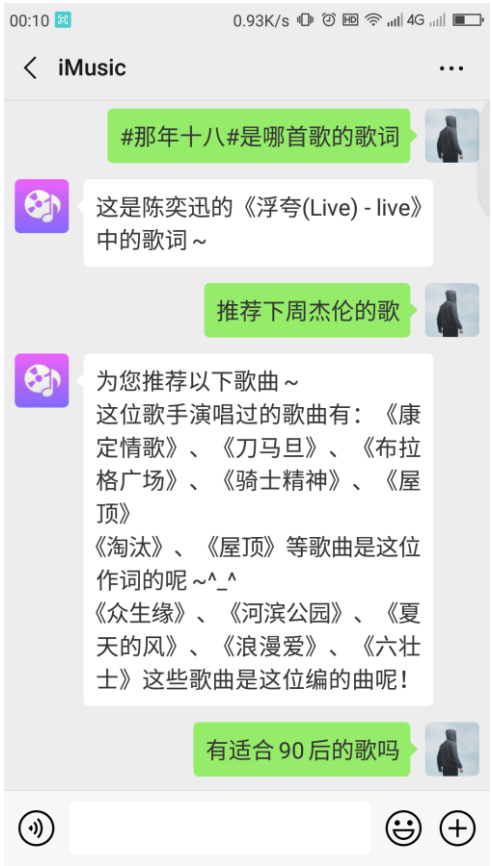
- We use the Wechat interface to provide the service, some screenshots:

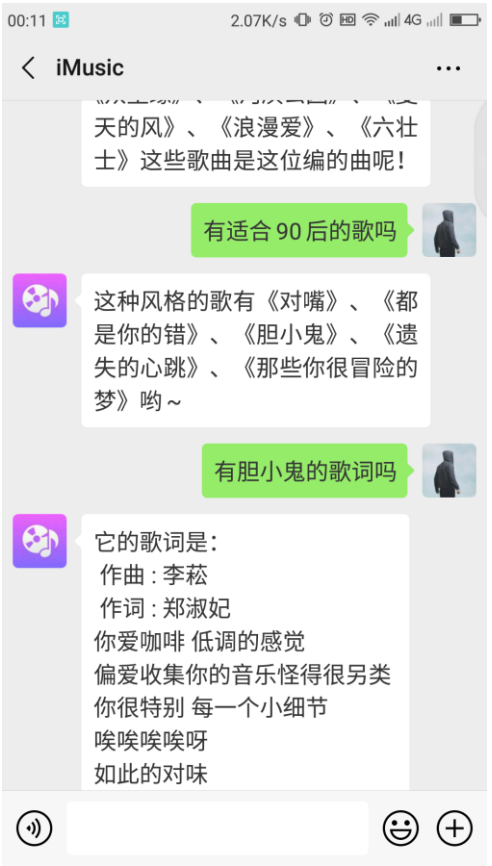




# Demo presentation

- We use the Wechat interface to provide the service, some screenshots:

# Evaluation on Recommendation System

- Three volunteers were invited to evaluate our recommendation system. The score ranks from 0 to 3 for each utterance, where 3 means the response satisfies the user's expectation while 0 is not.
- Each volunteer interacted to the system with 40 turns in total and evaluated it from both music-recommendation and chatting aspects (each aspect 20 turns). They finally also gave scores on the whole interaction process.

| | Avg_music | Avg_chat | Whl_music | Whl_chat |
|---|---|---|---|---|
| Evaluator#1 | 1.35 | 0.65 | 2 | 0 |
| Evaluator#2 | | | | |
| Evaluator#3 | | | | |

# Challenge

- Some challenges
  - Evaluation for recommendation performance, we have no ground truths;
  - Because the analysis for user intent is based on pre-designed patterns, some complex context will cause some problems.
  - The chat module performance is poor because of the lack of publicly high-quality Chinese dialogue corpus.
  - Due to time constraint, the chat platform cannot provide service for multi-users at same time, we will complete this point in future.
  - The current recommendation system is mostly based on text analysis and user preference, dealing with audio information and mining their relevance is much harder than language processing.

# Division of labor

- *Jia Chen： Whole system framework and algorithm design, dialogue data collecting and refining, implement the search module ($M_1$) and its patterns, passive recommendation module ($M_2$) and chat module ($M_4$), chat module evaluation and testing;*
- *Leilan Zhang: Crawl music data and dialogue data, analyze music data (LDA), test the chat module, implement music intent detection module ($M_0$) and initiative recommendation module ($M_3$) , integrate four modules together and connect the Wechat interface.*

***Thank you!***

***Q & A***