

Generating AIML Rules from Twitter Conversations

Hiroshi Yamaguchi, Maxim Mozgovoy

The University of Aizu

Tsuruga, Ikki-machi, Aizu-Wakamatsu, Fukushima, 965-8580 Japan

{m5201115, mozgovoy}@u-aizu.ac.jp

Abstract—A chat dialogue system or a conversational agent is a computer program designed to hold a conversation using natural language. Many popular chat dialogue systems are based on handcrafted rules, written in Artificial Intelligence Markup Language (AIML). However, manual design of rules requires significant efforts, so certain approaches for automating this process can be helpful. This paper presents some preliminary experiments to generate AIML knowledge automatically using conversation data acquired from Twitter. The experimental results show the possibility of obtaining natural-language conversation between the user and a dialogue system without the necessity of handcrafting its knowledgebase.

I. INTRODUCTION

A *chat dialogue system* or a *conversational agent* is a computer program designed to imitate a conversation with users [1]. In general, chat dialogue systems can be categorized into two types: *task-oriented* systems that are used to help the user to complete various tasks, and *open-domain* systems that aim to perform a natural conversation with the user [2].

A large number of dialog systems are rule-based, and one of the most popular mechanisms of representing rules is AIML (Artificial Intelligence Markup Language). AIML is a simple XML-based markup language that gained popularity after being used in a successful dialog system A.L.I.C.E. [3] that won the Loebner Prize three times. The main drawback of AIML-based systems lies in the large number of rules required to imitate a natural conversation, especially in case of open-domain systems. Therefore, AIML-based dialogue system require a lot of manual effort to describe its knowledgebase, leading to expensive and error-prone development process.

The goal of the proposed research is to demonstrate the possibility of generating AIML rules from Twitter conversations of real users. This method provides a potential possibility to greatly reduce the amount of manual effort needed to design an open-domain dialog system. We discuss the steps of this process, from gathering of the initial textual corpus to the operation of the resulting system, and provide example conversations.

Our ultimate aim is to develop a dialogue system supporting natural human-computer conversation, which can be useful in a variety of applications. The AIML-based approach is much simpler comparing to more advanced dialog system architectures [2]. Yet rule-base systems were still able to win the Loebner Prize, which proves their potential as human-like conversation agents. One of the main drawbacks of rule-based systems lies in the great number of rules that need to be handcrafted. We want to show that it is possible to reduce

the amount of human effort by automating rule generation process, using a large dataset of real human conversations, such as Twitter dialogs.

II. ELEMENTS OF AIML

As mentioned in the previous section, AIML is based on XML, and thus consists of hierarchically organized elements. The following explanations will include fragments of Japanese-language conversations since we relied on a Japanese Twitter corpus in the experiments.

A. AIML Categories

Individual “units of knowledge” are known as *categories* in AIML. Each category should define at least two compulsory elements: a *pattern* that contains a sample input, and a *template* that contains the corresponding response of the bot [4]. There are three types of categories in AIML: *atomic categories*, *default categories*, and *recursive categories* [1].

Atomic categories contain patterns without wildcard symbols *, ^, _ and #. These categories describe exact responses for precisely defined user input:

```
<category>
  <pattern>おはよう</pattern>
  <template>おはようございます。</template>
</category>
```

Here if the user inputs おはよう (“morning”), the bot should reply おはようございます (“good morning”).

Default categories contain patterns with some wildcard symbols. These patterns can be matched against a family of input strings and thus can be used to describe more universal conversational elements. For example, the wildcard character * can be used to match one or more input words as follows:

```
<category>
  <pattern>*は好きですか</pattern>
  <template>好きですね。</template>
</category>
```

Here if the user input anything that ends with は好きですか (“do you like *smth*”), the bot should reply 好きですね. (“I like it.”) The words matched with the wildcard characters can be programmed to appear in the bot response as well.

Recursive categories are used to handle certain patterns according to other rules already present in the system. In these categories, a <srail> tag is used to redirect the rule handler to another rule:

```
<category>
  <pattern>アケマシテオメデトウゴザイマス
```

```

</pattern>
<template>あけましておめでとうございます!!!
</template>
</category>
<category>
<pattern>アケマシテオメデトウゴザイマス
      コトシ モ ヨロシクオネガイシマス
</pattern>
<template><srai>アケマシテオメデトウ
ゴザイマス</srai></template>
</category>

```

The fragment above says that the user input アケマシテオメデトウゴザイマス コトシ モ ヨロシクオネガイシマス (“Happy New Year”) should be handled in the same way as アケマシテオメデトウゴザイマス (alternative way of wishing a happy New Year), thus the content of the tag <srai> in the second category matches the content of the tag <pattern> in the first category. Recursive categories are typically used to handle synonyms and complex grammatical forms by splitting them into simpler parts.

B. Interpreting AIML

The set of AIML rules forming a chatbot’s knowledgebase is processed with an *AIML interpreter*, responsible for actual dialogs with the user. There are several AIML interpreters available. We use Python-based pyAIML system, aimed to be fully compliant with AIML 1.0.1 standard [5].

AIML interpreter selects the rule that provides the longest match for the user input [1]. One can fine-tune rule preferences by using wildcards in the rules, since different wildcards have different priorities.

III. GENERATING AIML RULES

A. Twitter Streaming API

The Internet contains numerous samples of conversation data. Perhaps, Twitter dialogs come relatively close to informal daily conversations, and there is an API available to retrieve individual tweets and tweet streams [6].

We used Twitter API to create the “Utterance Pairs” database, consisting of individual utterances and replies to them. The collected data corresponds to the tweets posted from October 2016 to April 2017. To build the database, more than 70 thousands utterance pairs were collected using Twitter Streaming API [7] that allows to extract tweets in real time. Individual tweets are tagged with a number of attributes, including the tweet language, timestamp, and in-reply-to field. We fetched Japanese tweets only, and extracted the content of each attribute.

B. Converting Tweets to Rules

The process of converting the raw tweet dataset into a set of AIML rules consists of the following steps.

1) *Preprocessing*: Raw tweet data contains numerous messages without any conversational meaning that have to be considered garbage for our purposes. These tweets typically consist of hyperlinks, hashtags and user names, or contain no Japanese characters. We remove such tweets from the source collection. We also remove very long messages, containing over 30 Japanese characters. In general, noise filtering in Twitter data is a well-researched problem [8], and we are planning to extend our filtering algorithm in the future.

2) *Normalization*: Each utterance pair consists of the input sentence and the reply to the input sentence. These pair elements will be mapped to the AIML tags <pattern> and <template>. Since the pattern has to be matched against the user input during the actual user-bot interaction, we increase the chances of a successful match by *normalizing* the patterns. A typical normalization procedure for English consists of stemming and capitalization of the pattern. For the tweets in Japanese, we tokenize the input into individual morphemes using a morphological analyzer MeCab [9] convert them to katakana. We also use MeCab to perform a part-of-speech tagging of the input text and eliminate non-significant morphemes, such as auxiliary verbs, postpositional particles, conjunctions, and pre-noun adjectivals.

3) *Rule Generation*: Unknown words that cannot be recognized by MeCab are substituted with wildcards, thus forming the patterns of default category rules. All other patterns are transformed into the rules of atomic category. As a result, we obtain the rules of the following kind:

```

<category>
  <pattern>オハヨウ</pattern>
  <template>おはようございます(>_<)
</template>
</category>

```

IV. RESULTS

There are 71179 utterance pairs in the resulting dataset. After normalization and filtering, our program has generated 30991 atomic category rules and 39308 default category rules. By using the rules, our AIML bot is able to converse with the user, but the dialogs often contain unnatural responses, as the examples below show:

```

Conversation 1:
USER: こんにちは (Hello)
BOT: こんにちは!!...君に会えるなんて今日はきっと良い日だね (Hello, today is a good day to see you)
USER: それはよかった (It's good.)
BOT: 最高ですな。 (It's really good.)
USER: 何かいいことあった? (Any good news today?)
BOT: はるる (haruru)
USER: 好きな人でもできたの? (Do you have a crush on someone?)
BOT: めちゃドン引きやん? (笑) (Does something puts you off?)
USER: いや、別に引いてないけど (笑) (No, nothing)

```

BOT: べつにwww (Nothing.)

Conversation 2:

USER: おっはー (Good Morning)

BOT: おっはー! (Good Morning!)

USER: 今日めっちゃ暑くなるらしいよ (It seems that it will be really hot today)

BOT: えー (Oh-)

USER: 早く秋になってほしいね (I hope the season will change to fall early)

BOT: 仕方がない.....コミケあったし..... (It can not be helped... There was Comic Market...)

USER: コミケとか行ったことないわ (I have never been to Comic Market)

BOT: はい! コミケです! (Yes, Comic Market!)

USER: コミケとか絶対行きたくないわ (I really don't want to go to Comic Market)

BOT: はい! コミケです! (Yes, Comic Market!)

USER: アニメ好き? (Do you like animation?)

BOT: 追加して (Please add it)

V. DISCUSSION AND CONCLUSION

In this paper, we have presented a simple chatbot based on the AIML knowledge, extracted from Twitter streams. The bot is able to hold a conversation with the user, but many parts of the dialogs do not sound natural, and should be improved.

Probably, the largest shortcoming of the current implementation is related to the lack of context during conversations. The present bot implementation is based on the database of utterance pairs, while real dialogs include longer series of replies. AIML allows to incorporate context information in <that> tags, and we are planning to use this capability, and to substitute utterance pairs with longer dialog sequences.

Another difficulty is caused with the lack of approximate matching support in AIML. The only mechanism of approximate matching is wildcard characters, so the user-input

strings are expected to be precisely specified, or, at least, to follow well-defined grammatical patterns. Without wildcards, even a relatively large dataset like ours can be insufficient for providing an exact match in many cases, and thus lead to unnatural conversations (we currently generate a random utterance from the knowledgebase if no match is found).

However, we are planning to address this problem in the next revision of the system. Our present goal was to show that Twitter data can be indeed used as a basis for automated AIML rules generation, and the current system demonstrates that this work can be accomplished, and the proposed approach has a potential to assist labor-intensive manual design of chatbot knowledge.

REFERENCES

- [1] B. A. Shawar and E. Atwell, "ALICE chatbot: trials and outputs," *Computación y Sistemas*, vol. 19, no. 4, pp. 625–632, 2015.
- [2] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, and Y. Matsuo, "Towards an open-domain conversational system fully based on natural language processing," in *COLING*, pp. 928–939, 2014.
- [3] B. A. Shawar and E. Atwell, *A comparison between ALICE and Elizabeth chatbot systems*. University of Leeds, School of Computing research report 2002.19, 2002.
- [4] R. Wallace, "The elements of AIML style," *Alice AI Foundation*, 2003.
- [5] C. Stratton, "PyAIML: A Python AIML interpreter," <https://github.com/andelf/PyAIML>.
- [6] F. Bessho, T. Harada, and Y. Kuniyoshi, "Dialog system using real-time crowdsourcing and twitter large-scale corpus," in *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 227–231, Association for Computational Linguistics, 2012.
- [7] Twitter, Inc., "Twitter Streaming API," <https://dev.twitter.com/streaming/overview>.
- [8] R. Higashinaka, N. Kobayashi, T. Hirano, C. Miyazaki, T. Meguro, T. Makino, and Y. Matsuo, "Syntactic filtering and content-based retrieval of twitter sentences for the generation of system utterances in dialogue systems," in *Situated Dialog in Speech-Based Human-Computer Interaction*, pp. 15–26, Springer, 2016.
- [9] T. Kudo, "Mecab: Yet another part-of-speech and morphological analyzer," <http://mecab.sourceforge.net/>, 2005.