Metaphor, Humor and Emotion Processing in Human-Computer Interaction

Pawel Dybala[1], Michal Ptaszynski[2], Rafal Rzepka[3], Kenji Araki[3], Kohichi Sayama[4]
[1]JSPS Research Fellow / Otaru University of Commerce
Midori 3-5-21047-8501 Otaru, Japan
[2] JSPS Research Fellow / Hokkai-Gakuen University, Toyohira
Asahimachi 4-1-40, 062-0911 Sapporo, Japan
[3]Graduate School of Information Science and Technology
Hokkaido University, Kita 14 Nishi 9, Kita-ku
060-0814 Sapporo, Japan
[4]Otaru University of Commerce
Department of Information and Management Science
Midori 3-5-21, 047-8501 Otaru, Japan
{paweldybala, sayama}@res.otaru-uc.ac.jp, {kabura, araki, ptaszynski}@media.eng.hokudai.ac.jp

ABSTRACT: *In this paper we present a concept of a non-task oriented conversational system (chatterbot) for Japanese, able to generate humorous utterances in response to users' negative moods. Humorous utterances are either jokes (puns) or humorous metaphor misunderstandings. The system first detects user emotions from texts, and if they are negative, it tries to use humor to turn them into positive. Such an approach was tested in one of our previous works. Next the system uses one of the humor modules to generate a humorous response. User's response to this is again analysed to check his/her emotivereaction, and on this basis the system builds an individual model of his/her preferences. In this work we describe the components of the system, its general algorithm and resources necessary to construct it.*

Keywords: Text Processing, Emotion Detection from Text, Human-computer Interaction, Word Processing, Genetic Algorithm

Received: 11 November 2012, Revised 30 December 2012, Accepted 8 January 2013

1. Introduction

Today many experts in the fields of NLP and HCI agree that it is problematic to construct human-like computer systems without taking into consideration such ambivalent factors of human behaviour as humor or emotions. This was experimentally showed in numerous works (e.g. [1, 2]. Also in our research so far (summarized in [3, 4]) we showed that implementation of pun generator into a non-task oriented conversational system (chatterbot) can significantly improve its performance. We also showed that humor generation can be more effective when used in reaction to user'semotions. These were detected by an emotion-from-text analysis system.

In our current research project we decided to add a new element to the proposed system, which is humorous metaphor misunderstandings generation. To our best knowledge, no system that performs similar roles as the one described here has been constructed so far. Numerous research projects exist in humor, emotion and metaphor processing, none of which, however, joins these three areas. Our project is an interdisciplinary venture, that bridges them in order to construct a system that, as we believe, will be able to perform more natural linguistic interactions with users.

The project was launched recently. Its development steps and concepts of some of the system's components were mentioned in our recent works [5,6]. In this paper we introduce an overall and thorough description of the proposed system's algorithm as a whole, including a procedure for individualization of interaction (the individualization module). We also describe how the system's components interact with each other and how these interactions can be modified according to users‘ preferences.

Below we first briefly discuss some relevant works in the fields of humor processing (2.1) and metaphor processing (2.2). Next, we give the general outline of our system (3.1) and describe its components (modules): the decision module (3.2), the chatterbot module (3.3), the humor module (3.4), with its two sub-modules: the pun module and the metaphor module, and the individualization module (3.5). Finally we conclude the paper and give some further ideas concerning the system's development (4).

2. Relevant Works

In this section we briefly summarize relevant works in the areas of humor processing and metaphor processing. To the authors' best knowledge, no work exists in the area of HCI that would join these two topics. One that does so in the field of cognitive science is the work of Shen and Engelmayer, summarized in 3.4.

2.1 Humor processing

In the field of computational humor, the most popular genre is puns (or word plays), in which ambiguity is based on linguistic features, like homophony or polysemy. This makes puns more computable using methods and tools developed in the field of NLP (Natural Language Processing) - e.g. [7, 8, 9].

Many existing systems are of a stand-alone type, i.e. they generate humor in isolated forms, without integrating them into wider contexts. Other projects aim to integrate humor into human-computer interactions. Below we give a brief summary of state of the art in this field.

One of the best known of all works in the field of humor generation is JAPE punning riddles generator, developed by Binsted [7]. JAPE uses a set of symbolic rules and a large natural language lexicon to generate punning riddles, such as:

- What do you call a murderer with fiber?
- A cereal killer.

Riddles generated by the system were evaluated in an experiment in which schoolchildren assessed riddles generated by the system as being similarly funny as those created by humans.

JAPE system was implemented by Loehr [10] into a conversational agent Elmo "*living*" in a virtual game environment. However, experiments showed that relevant incorporation of punning riddles into a dialogue is quite problematic [10].

Another attempt of implementing JAPE generator into a system that interact with humans was made by Ritchie et al. [11], who developed a software that helps children with complex communication needs (CCN) improve their linguistic skills. The authors improved the riddle generation algorithm, and added a user interface. The system, named STANDUP, is described by the authors as "*a language playground, with which a child can explore sounds and meanings by making up jokes, with computer assistance* [11]." Experiments showed that the system received positive evaluation from the participants (children with CCN), and led to a significant increase in their communication skills.

The work of Ritchie et al. is of high importance in the field of humor-equipped systems, as it places pun generation in actual human-computer interaction. Moreover, the system received generally positive comments from the users, which shows that enhancing HCI with humor is a worthwhile enterprise.

Another worth mentioning work on pun generation is McKay's WISCRAIC system [8], which generates simple idiombased witticisms, such as:

"*The friendly gardener had thyme for the woman*!"

(thyme – type of a plant; homophone for the word "*time*" in the idiom "*have* time for someone).

The output is quite interesting and, most importantly, it is also generated according to the context (not in a dialogue, but inside the sentence). Unfortunately, to our knowledge, the WISCRAIC generator has yet to be implemented into an agent that would actually interact with humans, making assessment of its performance currently impossible [8].

Another attempt at implementing a joke generator into a conversational system was made by Tinholt and Nijholt [9], who constructed a cross-reference joke generator and combined it with a chatterbot.

The system's input was one sentence (utterance), towards which the generator produces humorous misunderstandings. For example, to a user's utterance:

User: Did you know that the cops arrested the demonstrators because they were violent?, the system can generate a quasi-misunderstanding response:

System: The cops were violent? Or the demonstrators? :)

The system (chatterbot with implemented joke generator) was evaluated in an automatic experiment, in which it analysed several chat transcripts and a simple story text. Unfortunately, cross-reference ambiguity occurs very rarely in real-life conversations, which, according to the authors, made it impossible to evaluate the system in conversations with human interlocutors [9].

All works mentioned above focused on constructing humor generators. However, also other studies exist, in which humorous contents are fully or partially preprogrammed. Although such works do not contribute directly to the field of computational humor, they are also of high importance in the field of enhancing HCI with humor.

An example of such study is one conducted by Augello et al. [12], who implemented a joke-telling algorithm into a conversational system. The jokes were not generated, but selected from a hand-made database. During conversations, the system first asked if the user would like to hear a joke and what it should be about and then it selected a proper one from a database.

Such a setup is quite natural in Western cultures, where jokes are often announced before they are told – e. g. "*I know a good joke, do you want to hear it*?" This mechanism, however, does not work in Japanese, which is the language we are dealing with in our research. The puns in Japanese (called dajare) are usually inserted naturally into a conversation, in order to surprise the listener. Thus, the setup proposed by Augello et al. would presumably not be appropriate to Japanese.

Another work in which preprogrammed humor was used to enhance HCI was done by Morkes et al. [13]. The authors placed jokes prepared beforehand in task-oriented conversations between humans and a conversational system. Experiments showed that the system that used humor was evaluated as more sociable, likeable and easier to cooperate with than a similar one without humor.

As showed above, the role of humor, be it generated or preprogrammed, is not an entirely neglected issue in HCI. However, to our knowledge, no existing research unites the two fields of humor processing and metaphor processing. Consequently, no dialogue system has been constructed that would be able to generate humorous metaphors. Also, no system that we know generates humorous contents accordingly to user emotions (see below).

This work combines the areas of humor, metaphor and emotion processing. The proposed algorithm should allow the system to use humor accordingly to user emotions, where by "*use humor*" we understand either humorous metaphor generation or simple pun generation (based on our previous work – [3, 4]).

2.2 Metaphor processing

2.2.1 State of the art

A good summary of current state of the art in metaphor processing is given by Shutova [14]. Here we only briefly summarize some of existing research in this field.

Works on automatic metaphor processing usually focus on either metaphor recognition or interpretation. Projects that belong to the former group deal with distinguishing metaphors from other texts. This includes detecting linguistic cues that indicate that the particular text is a metaphor, such as “*metaphorically speaking*” or “*so to speak*” (identified by Goatly [15]). Although detecting such expressions is not sufficient to identify metaphors, they can be used in more complex recognition algorithms.

Numerous attempts were made to recognize metaphors with the use of WordNet [16]. Among these we can name the work of Peters and Peters [17], Mason [18] or Krishnakumaran and Zhu [19]. Other works, like one by Gedigan et al. [20] use FrameNet [21] to acquire lexical data related to particular frames (“*motion*” and “*cure*”). Other works use large scale corpora to extract relations between source and target attributes.

One of the most common problems in metaphor recognition is caused by the fact that metaphors cannot be easily distinguished from other linguistic phenomena, such as polysemy. Therefore, many works stops at the distinction between “*literal*” and “*nonliteral*”.

As mentioned above, numerous works exist that attempt to deal with the issue of metaphor understanding. Martin’s MIDAS system [22] uses an existing metaphor database to query metaphors similar to inputted, and, if none such metaphor is found, it performs an ontological analysis to find analogical relations on a higher hierarchical level. This work is important also due to the fact that it was integrated with the Unix Consultant system, which is one of the few attempts to incorporate metaphor processing into a human-computer dialogue.

In other existing projects authors proposed metaphor-based reasoning frameworks, using manually coded knowledge about domains and concepts. One such system was KARMA, developed by Narayanan [23]. The system takes parsed text as input and operates mostly within the source domain. The results are next projected on a target domain.

Also worth mentioning is the work of Veale and Hao [24]. The authors proposed sets of characteristic concepts belonging to source and target domains. This knowledge was automatically extracted from the Internet. The sets are organized in a framework, in which operations like insertions or substitutions are performed to establish links between concepts. Unfortunately, it is still unclear, to what extent this approach is useful to interpret metaphors occurring in text.

2.2.2 MURASAKI Word Sense Description System

In above section we mentioned some metaphor recognising and metaphor understanding systems. However, as we are going to develop a system able to extend concept salience calculation so that it covers also humorous metaphors, we need an algorithm that would be able to recognize and interpret metaphors, as well as to provide us with a sort of concepts properties ranking. This humorous metaphors related project is conducted in Japanese (as our previous works). Therefore, we ideally need a system that would generate word descriptions for Japanese, and rank them according to their relation to the word or phrase.

A system that perform similar tasks is Masui et al.’s MURASAKI [25]. The system uses Internet query engines, such as Yahoo, to extract associations (descriptions) towards inputted word or phrase, and sorts generated associations from most to least plausible. This is done by checking co-occurrences of each description with the input. On this basis, score is calculated that reflects the description’s appropriateness (i.e. degree of relevance to the inputted phrase). Thus, it can be stated that MURASAKI system calculates salience of concept properties. For instance, towards the word *ringo* (apple), the system would generate a list of descriptions like one below:

1. *kaori* (aroma) score: 0.147
2. *sawayaka* (invigorating) score: 0.075
3. *fruity* (fruity) score: 0.062
4. *sanmi* (sourness) score: 0.053

. ...

10. *hoo* (cheeks) score: 0.026

The details of score calculation can be found in [25].

3. System Outline

In this section we propose an idea of a conversational system for Japanese, able to use humor in reaction to users’ emotions.

Humor can be generated by one of two modules: pun generation module and metaphor misunderstanding generation module. The system also uses an emotiveness analysis module to detect whether humor should be used or not. This module is also used to analyse users' utterances, on which basis each particular user's preference model can be built, which should allow the system to adapt to every user's individual needs.

In this section we first describe the system's general algorithm and its modules.

3.1 General algorithm
Figure 1 shows the outline of proposed system's algorithm. The system is a chatterbot (non-task oriented conversational system), able to perform free conversations with users. The system and its all components process text based input in Japanese..

The input of the system is user's utterance (marked as "*Utterance* 1" on Figure 1). It is first analysed by the decision module (3.2), whose central component is an emotiveness analysis system. It detects textual expression of emotions in user utterances, and decides whether they are positive, negative or neutral. Next, on the basis of the outcome of this analysis, the module decides if humor should be generated to make this user feel better (or less negative). If the decision module decides that no humor should be used, a non-humorous response is generated by the chatterbot module (3.3). If the decision module decides that humor should be used, a humorous response is generated by the humor module (3.4). It contains of two sub-modules: the metaphor module and the pun module. The former generates humorous metaphor misunderstandings and the latter generates simple puns. As opportunities of generating metaphor misunderstandings in occur less frequently in daily language, in the initial version of the system, we are planning to give priority to the metaphor module.

The system first checks if the Utterance 1 contains a metaphor (which is a necessary condition to generate a misunderstanding). If yes, it tries to generate its humorous misunderstanding. If no, or if the metaphor module fails to generate a misunderstanding, a humorous response is generated by the pun module.

Let us assume that to the response (be it humorous or nonhumorous) generated by the system, the user reacts with another utterance (marked as "*Utterance* 2" on Figure 1). This is again processed by the system to continue the conversation. However, all user utterances are also processed by the individualization module (3.5). Its main task is to gather data regarding each particular user's preferences in order to build his/her individual model of humor sense. The main component of the individualization module is again the emotiveness analysis system, which detects textual expression of emotions.

As showed on the example in Figure 1, the emotiveness analysis of user's utterances can provide us with such data as: what type of humor he/she prefers, towards what moods should the system use humor, and when is it better to avoid it. The more such data the system collects, the better it can adapt to each particular user's needs.

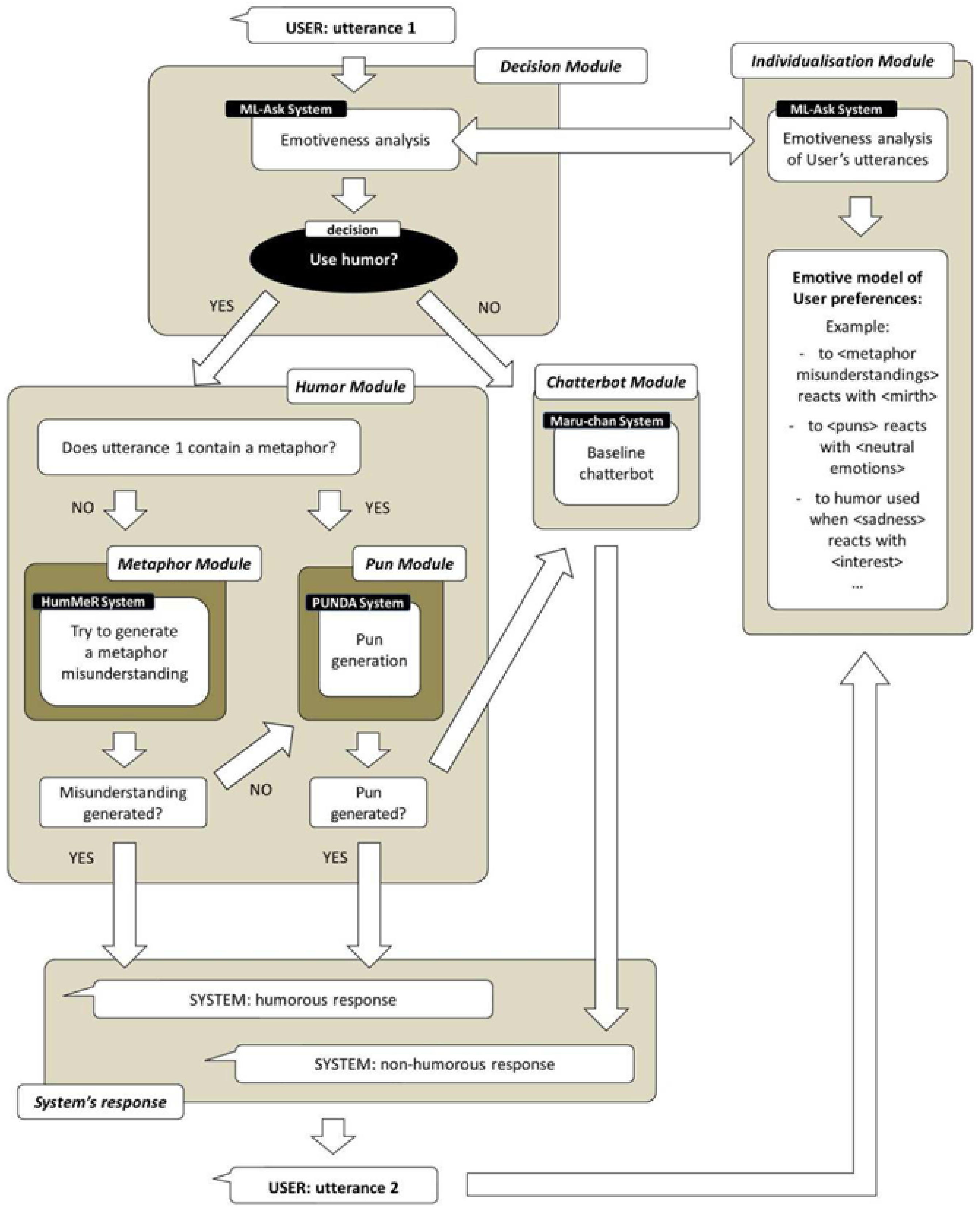
3.2 Decision module
As mentioned above, user's utterance is first analysed by the decision module in order to decide if the system's response should be generated by the humor module or by the chatterbot module. This decision is made on the basis of emotiveness analysis, performed by an emotiveness analysis system for Japanese.

The system, named ML-Ask, was developed in our previous research (see [26] for details). Its algorithm outline is presented on Figure 2. The system first analyses the inputted sentence to check its emotiveness. This is done by checking if it contains socalled "*emotive elements*" (elements that do not convey any particular emotions, but make the sentence emotive in general). For example, the sentence:

"*Kono hon saa, sugee kowakatta yo. Maji kowasugi*!"
(That book, ya know, 'twas a total killer. It was just too scary.)

, is recognized as emotive, as it contains emotive elements: *saa* (emphasis), *sugee* (totally), *yo* (emphasis), *maji* (really), -*sugi* (too much) and an exclamation mark.

If the sentence was recognized as emotive, the system next detects emotion types it contains. This is done by checking if the sentence contains any "*emotive expressions*", i.e. expressions that convey particular emotions. To do this, the system uses a data base of Japanese emotive expressions, based on Nakamura's Emotive Expressions Dictionary. For example, in the sentence above, the system found the emotive expression *kowai* (scary), which belongs to the group called *kyoufu* (fear).



USER: utterance 1
Decision Module
ML-Ask System
Emotiveness analysis
decision
Use humor?
YES
NO
Individualisation Module
ML-Ask System
Emotiveness analysis of User's utterances
Emotive model of User preferences:
Example:
- to <metaphor misunderstandings> reacts with <mirth>
- to <puns> reacts with <neutral emotions>
- to humor used when <sadness> reacts with <interest>
...
Humor Module
Does utterance 1 contain a metaphor?
NO
YES
Metaphor Module
HumMeR System
Try to generate a metaphor misunderstanding
Pun Module
PUNDA System
Pun generation
Misunderstanding generated?
NO
Pun generated?
YES
YES
Chatterbot Module
Maru-chan System
Baseline chatterbot
SYSTEM: humorous response
SYSTEM: non-humorous response
System's response
USER: utterance 2

Figure 1. Proposed system's general algorithm outline

If no such expression is recognized, the system uses Shi et al.'s web-mining technique [27] to extract emotive associations from the Internet. The input is queried in Yahoo to check its emotive associations by counting which emotive expressions follow it most often.

As the result, we obtain an emotiveness analysis summary, such as below:

Sentence: *Kono hon saa, sugee kowakatta yo. Maji kowasugi*!" (That book, ya know, 'twas a total killer. It was just too scary.)
Emotive elements: *saa* (emphasis), *sugee* (totally), *yo* (emphasis), *maji* (really), *-sugi* (too much), exclamation mark
Emotive value: 6 (above zero -> specify types of emotions)
Emotive expressions: *kowai* (frightening)
Emotions found: *fear*
Valence: *negative*

Sentence: *Kyou wa atatakai desu ne*. (It's warm today, isn't it?)
Emotive elements: *-ne* (-isn't it)
Emotive value: 1 (above zero -> specify types of emotions)
Emotive expressions: none (-> use web mining procedure)
Emotions found on the Web: joy
Valence: positive

Performance of the ML-Ask system was tested in numerous evaluation experiments, which showed that it can successfully detect emotions from users utterances [7]. We also used it in our previous research on humor-equipped conversational systems [3, 4], in which it also proved useful and usable.

As stated above, the decision whether or not to use humor is made on the basis of the emotiveness analysis performed by MLAsk. In the initial settings of the system we are planning to employ a simple rule: if user's emotive state is negative or neutral – the response should be generated by the humor module. Otherwise, the response should be generated by the chatterbot module. This approach was already tested in our previous research (see [3]), which showed that humor-equipped system can make users feel better or at least reduce the negativity in their moods. However, since we are also planning to implement the individualization procedure (see 3.5), this setting can be changed for every particular user. If, for instance, the system states that one user does not like being told jokes when sad, it should modify the decision patterns to avoid eliciting negative emotions for this particular user.

3.3 Chatterbot module

If the decision module decides that the system should not use humor, the response to user's utterance is generated by the chatterbot module. Its central part is based on Takahashi's Maruchan conversational system for Japanese [28]. The system uses the Internet to extract word associations for users' utterances, and then uses them to generate a relevant response. For example, from the sentence: (There's something cold to drink in the fridge)

"*Reizouko no naka ni tsumetai nomimono ga arimasu*"
The system extracts keywords: "*reizouko*" (fridge)

"*tsumetai*" (cold) and "*nomimono*" (something to drink), and then queries them in the Yahoo search engine to extract word associations (nouns and adjectives with the highest occurrence frequency), which are next used to generate response candidates. For the above example, the response generated would be:

"*Tsumetai juusu nara sakki nomimashita yo*"
(If you mean cold juice, I have just drunk it!).

The systems was used and tested in our previous research. For more details, see [28].

3.4 Humor module

If the decision module decides that the system should use humor, the response to user's utterance is generated by the humor

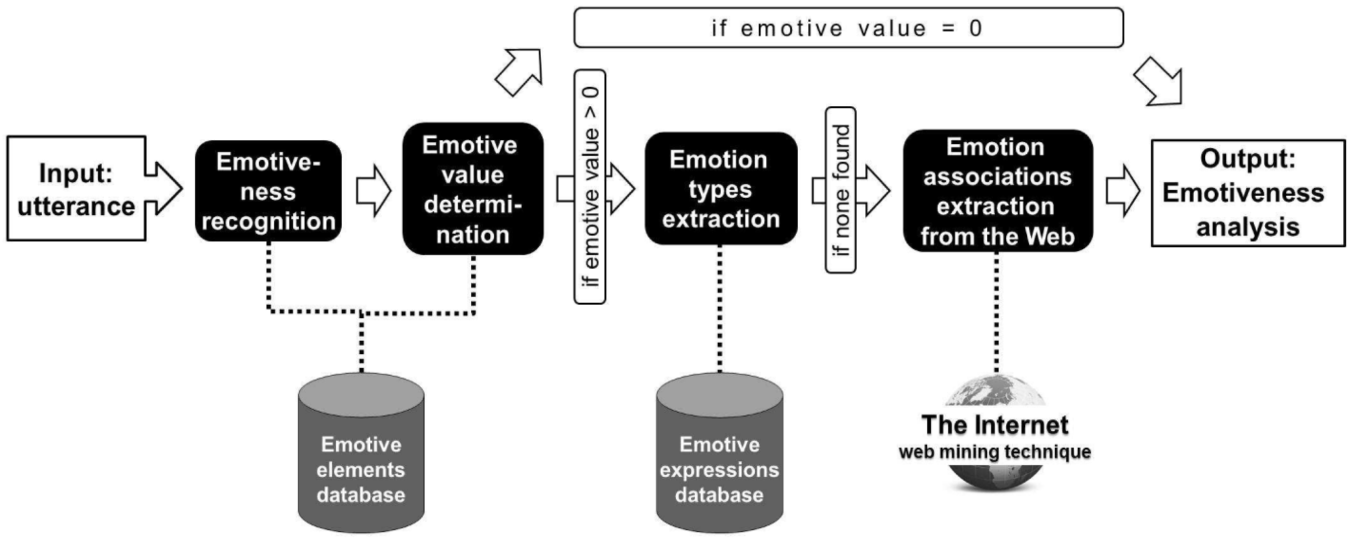

if emotive value = 0
Input: utterance
Emotive-ness recognition
Emotive value determi-nation
if emotive value > 0
Emotion types extraction
if none found
Emotion associations extraction from the Web
Output: Emotiveness analysis
Emotive elements database
Emotive expressions database
The Internet
web mining technique

Figure 2. ML-Ask System – algorithm outline

module, i.e. by one of its sub-modules: metaphor module and pun module. As mentioned above, in the initial settings the priority here is given to the former, as possibilities to generate humorous metaphor misunderstandings occur less often in daily conversations. This, however, can be changed according to user's preferences, defined by the individualization model (3.5).

By giving priority to the metaphor module we mean setting the humor module to check if user's utterance contains any metaphor. This is done by procedures described below (metaphor module). If the input contains metaphor, the system attempts to generate a humorous misunderstanding. Otherwise, the system tries to generate a pun.

3.4.1 Metaphor module

This model generates humorous metaphor misunderstandings. Its core part is the HumMeR System, a software which is currently under development [5]. Its algorithm is presented on Figure 3.

The system bases of the findings of Shen and Engelmayer [29] regarding differences in degrees of salience imbalance in humorous and non-humorous metaphors. Salience imbalance is a theory that explains mechanisms working in metaphors, proposed by Ortony [30]. It states that in metaphorical expressions certain highly salient properties of the metaphor source are matched with much less salient properties of metaphor target. In other words, certain properties of the target, which are normally perceived as not very salient, become more salient by comparing the common ground between the target and the source [30]. In metaphorical comparison like this: "*Billboards are like warts - they are ugly and stick out*", very salient properties of "*warts*", such as "*ugliness*" or "*sticking out*", are at the same time not very salient (albeit not completely implausible) properties of "*billboards*".

In their work, Shen and Engelmayer experimentally showed that the degree of salience imbalance is higher in humorous than in non-humorous metaphors (see [29] for details). They also showed that humorous metaphors often include what the authors call "*a shift in emotional load of the two concepts*" that constitute the metaphorical expression [29]. By this the authors understand that humoristic effect in metaphors can be enhanced or even co-produced by a discrepancy between emotional valence (positive, neutral or negative) of two concepts that constitute the metaphor. In our system development we used these two findings (increased salience imbalance degree and emotional shifts in humorous metaphors). The system's algorithm is described below.

The system first checks if the inputted utterance contains a known metaphor, i.e. such that can be found in our metaphor database (contains popular metaphors in Japanese). If yes, the system uses the known metaphor processing procedure. It first checks the metaphor's salience imbalance, using the metaphor salience database, and marks the degree as "*a*".

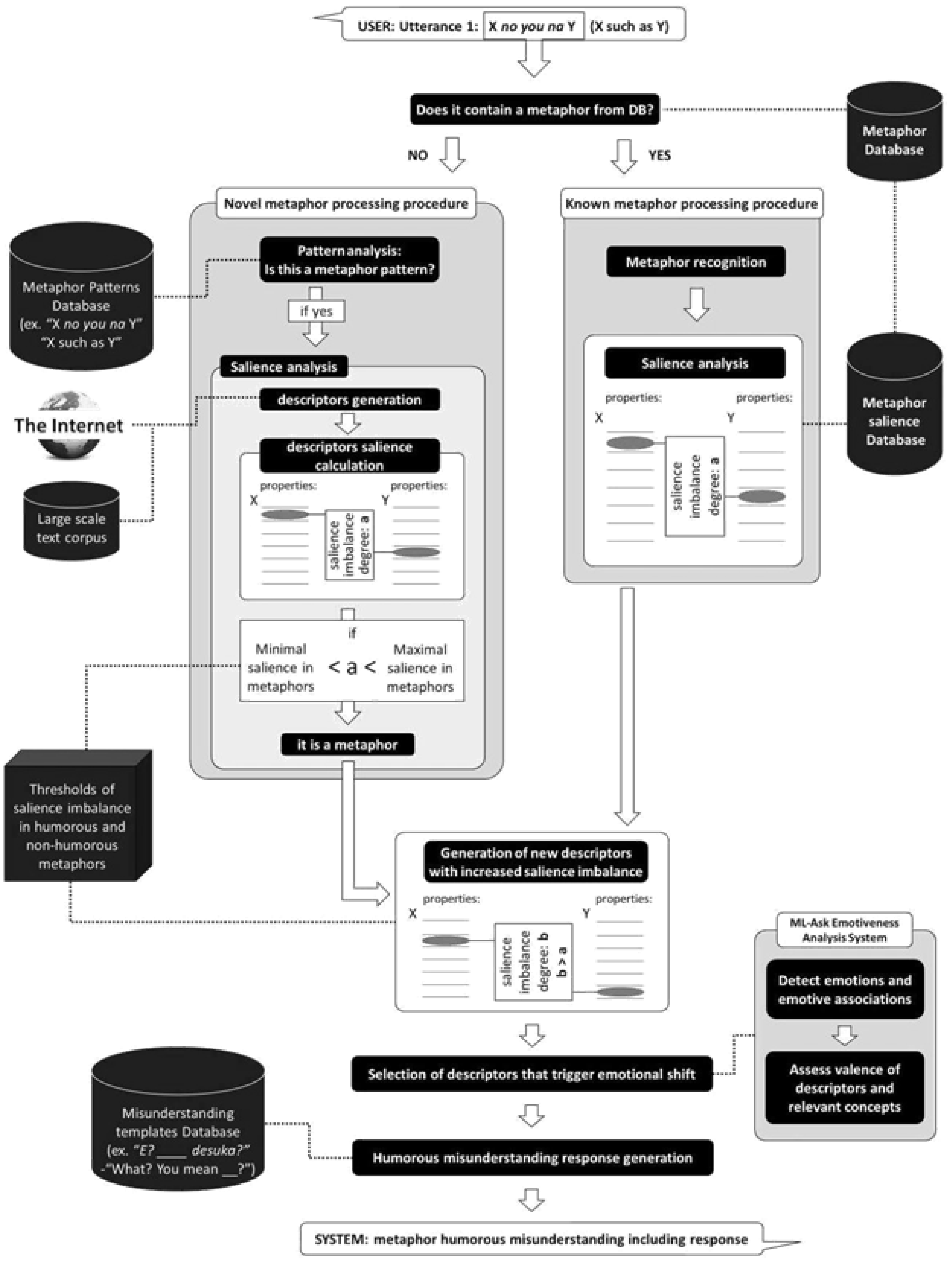

USER: Utterance 1: X no you na Y (X such as Y)
Does it contain a metaphor from DB?
Metaphor Database
NO
YES
Novel metaphor processing procedure
Known metaphor processing procedure
Metaphor Patterns Database (ex. "X no you na Y" "X such as Y"
Pattern analysis: Is this a metaphor pattern?
if yes
Metaphor recognition
Salience analysis
descriptors generation
The Internet
descriptors salience calculation
Large scale text corpus
properties:
X
Y
salience imbalance degree: a
Metaphor salience Database
if
Minimal salience in metaphors
< a <
Maximal salience in metaphors
it is a metaphor
Thresholds of salience imbalance in humorous and non-humorous metaphors
Generation of new descriptors with increased salience imbalance
salience imbalance degree: b b > a
ML-Ask Emotiveness Analysis System
Detect emotions and emotive associations
Assess valence of descriptors and relevant concepts
Selection of descriptors that trigger emotional shift
Misunderstanding templates Database (ex. "E? ____ desuka?" -"What? You mean __?")
Humorous misunderstanding response generation
SYSTEM: metaphor humorous misunderstanding including response

Figure 3. HumMeR System (humorous metaphor misunderstandings generator) – algorithm outline

If t   he input does not contain a known metaphor, it uses the novel metaphor processing procedure. First the system analyses it to check if it contains any of the metaphor patterns (such as, for instance, "*X no you na Y*" – "*X such as Y*"). To do that, it uses a metaphor patterns database. If such pattern is found, the system enters the next phase, in which it calculates salience of the components found in the input. The system first generates descriptors (properties) for each component, and calculates their salience, using similar procedures that those proposed by Masui et al. [25] in MURASAKI System (see 2.2.2). As the resource at this point, the system will use the Internet, although we are also planning to use a large scale text corpus [31], or a combination of these two. Next, the system compares the calculated salience of each descriptor and salience imbalance between them ("*a*") to a set of thresholds of salience in non-humorous metaphors[2] . If it is higher than the minimal and lower then maximal value of salience imbalance set for non-humorous metaphors, the input is classified as a metaphor.

The salience imbalance "*a*", calculated in the previous stage (either in the known or novel metaphors processing procedure), is next used as a baseline to extract another pair of descriptors with salience imbalance "*b*". The salience imbalance "*b*" should be higher than "*a*" (as explained above, humorous metaphors Have higher salience imbalance than non humorous ones). This is compared with thresholds of salience imbalance in humorous and non-humorous metaphors. If, for instance, recalculated salience imbalance "*b*" exceeds the maximal level defined for humorous metaphors, there is a possibility that the components of metaphor (new properties of the target and source) are too distant and thus constitute rather abstract than humorous sentence.

The newly extracted pair of descriptors, along with those used in the inputted metaphor, will then be analysed by the ML-Ask system to check their emotional valence. To do this, ML-Ask uses the web mining procedure to query the Internet for emotive associations, as described in 3.2. Next, the system checks the valence of extracted emotive associations in order to choose the description which valence is opposite to those in the input.

Finally, the system uses metaphor misunderstanding templates database in order to generate a humorous response to user's utterance.

For example, from a user's utterance:

"*My friend has legs like a deer*."

the system will first detect a metaphor present in the database, along with its descriptors, i.e. "*slim, nice*", and their salience imbalance "*a*". Next, the system will try to generate another pair of descriptors, common for both components ("*legs*" and "*deer*"), such as "*hairy*", whose salience imbalance "*b*" should be higher than "*a*". Next, each such candidate will be analysed by the MLAsk system to check if the emotional shift occurs in it. If the phrase "*hairy legs*" is found to be associated as not positive, contrary to the phrase "*slim legs*", the candidate can be used to generate a response like:

-You mean hairy?,

which will be done by using a metaphor misunderstanding template "*You mean* __?".

3.4.2 Pun module

The pun module generates punning responses to users' utterances. Its core part is PUNDA System, developed and experimentally tested in our previous research. A detailed description of this system can be found in [3, 4].

The pun module first extracts a base phrase (phrase that will be transformed into a pun) from the inputted utterance. It can be either a noun, an adjective or a verb. Next, the system uses pun generation patterns to generate phonetic candidates towards that base phrase. Then the system converts each candidate to Japanese ideograms (Kanji characters). Next it checks if any of converted phrases is an existing word. If no phrase is found to be an existing word, the system chooses a pun from a database." If yes, the system checks its hit rate on the Internet, and then chooses the one with the highest hit rate for the pun candidate, which is next integrated into a sentence using Japanese pun sentence templates, prepared beforehand.

Below we present an example of the system in action:

[2]Thresholds of salience imbalance will be acquired by analysing salience imbalance degrees in humorous and non-humorous metaphors.

User: -*Kaeru daikirai*! (I hate frogs!)

Phonetic candidates: *akaeru, ikaeru, ukaeru, tsukaeru....*

Candidate chosen: *tsukaeru* (be able to use)

Generation pattern used: initial mora addition

System: -*Kaeru to ieba tsukaeru no desu ne*. (Speaking of frogs, we could use that!)

This algorithm, albeit quite simple, was positively evaluated by users in numerous experiments (described in details in [3, 4]). In one of them they were asked to perform comparative evaluation of two chatterbots: which and without implemented PUNDA System. The former was assessed by most users as generally better, more friendly and making the users feel more positive than the latter. These results were confirmed in another experiments, in which we conducted an automatic emotiveness analysis of the conversation logs between users and systems. To do that, we used the ML-Ask System (see 3.2). The results of this analysis showed that the humor-equipped chatterbot elicited more emotions in users, and most of these emotions were positive.

3.5 Individualization module

Individualization is a crucial issue in HCI, especially in systems designed to interact with users on daily basis. Among chatterbot real life applications experts often name such products as car navigators, able to entertain drivers with chatting, or conversation companions for lonely or elderly people. Thus, systems designed to perform such roles should be able to adapt to each individual user's needs and preferences, i.e. to modify their performances accordingly. Therefore we decided to implement the individualization module into our system.

The individualization module is currently under construction [32]. Its core function is again performed by the ML-Ask System (see 3.2). Above we described ML-Ask's role in the decision module, in which it performs emotiveness analysis of user's utterances, and on this basis decides if the system should use humor. The outcome of this analysis is stored by the individualization module and used to construct each particular user's personal profile.

Also, user's responses to utterances generated by the system are analysed by ML-Ask to check their emotive reactions. These data are too stored by the individualization module. The more data the system acquires in this manner, the more complex and accurate the user's profile should be created.

An example of how such profile may look like is presented on Figure 1. If the user reacts with the positive emotion of mirth when told a metaphor misunderstanding, the system will store this message, and if the same user in another turn of conversation reacts less positively when told a pun, the system can assume that this particular user prefers the former than the latter. Also, while monitoring the interactions, the individualization module can detect correlations between emotions preceding and following humorous acts. If, for instance, the user reacts with positive emotion of interest when told a joke when he/she was sad, the system can assume that for this particular user this strategy of making the partner's feelings better is probably appropriate. On the other hand, for instance, if the user reacts with negative emotions, such as irritation, to humor used by the system towards emotions such as anger, the system can assume that it might be better to avoid such strategies.

The more data gathered by the individualization module for every particular user, the more accurate profiles it can built and the better it can adapt to their individual needs. Moreover, the fact that the conversations are constantly monitored by the MLAsk system can be used to gather also other information that can be used in the user profiles. This does not need to be limited to the use of humor. The individualization module can collect information regarding users' reactions towards particular conversation topics, styles, emotiveness of system's utterances and other conversation features. By doing so, the system will be able to chose the most appropriate conversation strategies for every user.

4. Conclusion and Future Work

In above sections we described a concept of a conversational system for Japanese, able to use humor during conversations with users. Being an extended version of a system developed in our earlier works (see [3, 4]), this system can generate two types of humorous contents: puns and humorous metaphor misunderstandings. The decision whether humor should be used is made on the basis of emotiveness analysis of user's utterance. Additionally, the proposed system will be able to adapt its performance to every user's personal needs, by constructing their individual profiles, also based on the outcome of the emotiveness analysis.

The project described here is still a work in progress. That said, there are some improvements we are planning to make after finishing the basic version of the system. We, for example, will enhance the salience calculation and recalculation procedures (described in section 3.4) by adding taxonomy based algorithms. By this we understand using such resources as WordNet [16] or *Bunrui goi hyou for Japanese* [33], from which the system will extract information about each concept's relations, which in turn will allow it to calculate salience imbalance also between their hypernyms. This will extend the system's possibilities of generating humorous metaphor misunderstandings, by rising the probability of finding a pair of descriptions that will constitute a misunderstanding considered as funny.

We are also planning to evaluate the system's performance in numerous experiments. To do that, we will use the methodology proposed and tested in our previous works [3,4].

5. Acknowledgements

This work was supported by KAKENHI (Project Number: 23-01348)

References

[1] Ritchie, G., Manurung, R., Pain, H., Waller, A., Black, R., O'Mara, D. (2007). A practical application of computational humour. *In*: 4th International Joint Conference on Computational Creativity, p. 91-98. London, UK.

[2] Morkes, J., Kernal, H. K., Nass, C. (1999). Effects of humor in taskoriented human-computer interaction and computer-mediated communication: A direct test of srct theory. *Human-Computer Interaction*, 14 (4) 395-435.

[3] Dybala, P. (2011). Humor to Facilitate HCI: Implementing a Japanese Pun Generator into a Non-task Oriented Conversational System, Lambert Academic Publishing.

[4] Dybala, P., Ptaszynski, M., Maciejewski, J., Takahashi, M., Rzepka, R., Araki, K. (2010). Multiagent system for joke generation: Humor and emotions combined in human-agent conversation. *In*: JAISE, Thematic Issue on Computational Modeling of Human-Oriented Knowledge within Ambient Intelligence, 2, 31-48.

[5] Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K., Sayama, K. (2012). Beyond Conventional Recognition: Concept of a Conversational System Utilizing Metaphor Misunderstanding as a Source of Humor. *In*: JSAI 2012, Alan Turing Year Special Session. Yamaguchi, Japan.

[6] Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K., Sayama, K. (2012). Emotion Valence Shifts in Humorous Metaphor Misunderstandings Generation. *In*: AISB/IACAP 2012 Symposium: LaCATODA 2012, p. 40-50. Birmingham, UK.

[7] Binsted, K. (1996). Machine humour: An implemented model of puns, Ph.D. Dissertation, University of Edinburgh, UK.

[8] McKay, J. (2002). Generation of idiom-based witticisms to aid second language learning, *In*: Stock et al., p. 77-87.

[9] Tinholt, H. W., Nijholt, A. Computational Humour: Utilizing Cross-Reference Ambiguity for Conversational Jokes. *In*: Proceedings of 7th International Workshop on Fuzzy Logic and Applications (WILF 2007), Camogli (Genova), Italy. LNAI Vol.4578. Springer Verlag, p. 477-483.

[10] Loehr, D. (1996). An integration of a pun generator with a natural language robot, *In*: Proceedings of the International Workshop on Computational Humor, J. Hulstijn, A. Nijholt, eds., University of Twente, Netherlands, p. 161-172.

[11] Ritchie, G., Manurung, R., Pain, H., Waller, A., Black, R., O'Mara, D. (2007). A practical application of computational humour, *In*: Proceedings of the 4th International Joint Conference on Computational Creativity, p. 91-98.

[12] Augello, A., Saccone, G., Gaglio, S., Pilato, G. (2008). Humorist Bot: Bringing Computational Humour in a Chat-Bot System, *In*: Proceedings of International Conference on Complex*, Intelligent and Software Intensive Systems* (CISIS 2008). Barcelona, Spain, p. 703-708.

[13] Morkes, J., Kernal, H. K., Nass, C. (1999). Effects of humor in taskoriented human-computer interaction and computer-mediated communication: A direct test of srct theory, *Human-Computer Interaction*, 14 (4) 395-435.

[14] Shutova, E. (2010). Models of Metaphor in NLP, *In*: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, p. 688–697, Uppsala, Sweden.

[15] Goatly, A. (1997). The Language of Metaphors, Routledge, London.

[16] Fellbaum, C. (ed.). (1998). WordNet: An Electronic Lexical Database. MIT Press.

[17] Peters, W., Peters, I. (2000). Lexicalised systematic polysemy in wordnet, *In*: Proceedings of LREC 2000, Athens.

[18] Mason, Z. J. (2004). Cormet: a computational, corpus-based conventional metaphor extraction system, *Computational Linguistics*, 30 (1) 23–44.

[19] Krishnakumaran, S., Zhu, X. (2007). Hunting elusive metaphors using lexical resources, *In*: Proceedings of the Workshop on Computational Approaches to Figurative Language, p. 13–20, Rochester, NY.

[20] Gedigian, M., Bryant, J., Narayanan, S., Ciric, B. (2006). Catching metaphors, *In*: Proceedings of the 3rd Workshop on Scalable Natural Language Understanding, p. 41–48, New York .

[21] Fillmore, C. J., Johnson, C. R., Petruck, M. R. L. (2003). Background to FrameNet, International Journal of Lexicography, 16 (3) 235– 250 .

[22] Martin, J. H. (1990). A Computational Model of Metaphor Interpretation, Academic Press Professional, Inc., San Diego, CA, USA.

[23] Narayanan, S. (1997). Knowledge-based action representations for metaphor and aspect (KARMA), PhD thesis, University of California at Berkeley.

[24] Veale, T., Hao, Y. (2008). A fluid knowledge representation for understanding and generating creative metaphors, *In*: Proceedings of COLING , p. 945–952, Manchester, UK.

[25] Masui, F., Kawamura, Y., Fukumoto, J. Isu, N. (2008). MURASAKI: Web-based Word Sense Description System. *In*: ITC-CSCC, p. 1285-1288. Shimonoseki, Japan.

[26] Ptaszynski, M., Emotion Awareness in Dialog Agents. (2011). Affect Analysis of Textual Input Utterance and its Application in Human-Computer Interaction. Lambert Academic Publishing.

[27] Shi, W., Rzepka, R., Araki, K. (2008). User Textual Input Using Causal Associations from the Internet, *In*: FIT2008, p. 267-268. Fukusawa, Japan.

[28] Takahashi, M. (2009). Web ni yoru kyoukihindo to n-gram moderu wo mochiita hatsuwabunnseiseishuhou (Utterance Generation Method Using Web Search Results and Word n-grams). Bachelor dissertation. Sapporo, Japan: Hokkaido University.

[29] Shen, Y., Engelmayer, G. (2012). A friend is like an anchor - sometimes you want to throw them out of the boat: What makes a metaphorical metaphorical comparison humorous?. Submitted for publication in Metaphor and Symbol, (available at: http://www.tau.ac.il/~yshen/publications/%20friend%20is%20like %20an%20anchor.pdf)

[30] Ortony, A. (1979). Beyond literal similarity, *Psychological Review*, 86 (1) 161-180.

[31] Maciejewski, J., Ptaszynski, M., Dybala, P. (2010). Developing a Large- Scale Corpus for Natural Language Processing and Emotion Processing Research in Japanese. *In*: IWMST, p.192-195. Kitami, Japan.

[32] Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K. (2009). Humorized Computational Intelligence - towards User-Adapted Systems with a Sense of Humor. *In*: Giacobini, M., Brabazon, A., Cagnoni et al. (eds.) EvoStar 2009 Conference, EvoWorkshops. LNCS, 5484, p. 452-461. Springer, Berlin & Heidelberg.

[33] National Institute for Japanese Language. (2004). Bunrui Goi Hyo: revised and enlarged edition Dainippon Tosho.

[34] Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K. (2010). Evaluating Subjective Aspects of HCI on an Example of a Non-Task Oriented Conversational System, *IJAIT*, 19 (6) 819–856. World Scientific Publishing Company.