Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2011, Article ID 107310, 12 pages
doi:10.1155/2011/107310

Research Article

Reducing Excessive Amounts of Data: Multiple Web Queries for Generation of Pun Candidates

Pawel Dybala,[1] Michal Ptaszynski,[2] and Kohichi Sayama[3]

[1] *Otaru University of Commerce, Midori 3-5-21, Otaru 047-8501, Japan*
[2] *High-Tech Research Center, Intelligent Techniques Laboratory 6, Hokkai-Gakuen University, Minami 26, Nishi 11, Chuo-ku, Sapporo 064-0926, Japan*
[3] *Department of Information and Management Science, Otaru University of Commerce, Midori 3-5-21, Otaru 047-8501, Japan*

Correspondence should be addressed to Pawel Dybala, paweldybala@kotoken.pl

Received 12 July 2011; Revised 25 November 2011; Accepted 6 December 2011

Academic Editor: Srinivas Bangalore

Humor processing is still a less studied issue, both in NLP and AI. In this paper we contribute to this field. In our previous research we showed that adding a simple pun generator to a chatterbot can significantly improve its performance. The pun generator we used generated only puns based on words (not phrases). In this paper we introduce the next stage of the system's development—an algorithm allowing generation of phrasal pun candidates. We show that by using only the Internet (without any hand-made humor-oriented lexicons), it is possible to generate puns based on complex phrases. As the output list is often excessively long, we also propose a method for reducing the number of candidates by comparing two web-query-based rankings. The evaluation experiment showed that the system achieved an accuracy of 72.5% for finding proper candidates in general, and the reduction method allowed us to significantly shorten the candidates list. The parameters of the reduction algorithm are variable, so that the balance between the number of candidates and the quality of output can be manipulated according to needs.

1. Introduction

The subfield of natural language processing (NLP) called "humor processing" is still quite neglected, and this paper can be seen as one attempt to bridge this gap. In this paper we first give some examples of the beneficial functions of humor (Section 1.1), then we explain the importance of humor processing, briefly summarize its state of the art (Section 1.2), we point out some possible applications (Section 1.3), give a brief explanation of Japanese homophony (Section 1.4), and introduce our contribution to the field (Section 1.5). In further sections, we describe our algorithms (Section 2) and experiments conducted to verify their usability (Section 3). Then we discuss the issue of candidates number (Section 4), analyze the results, and point out some directions for the future (Section 5).

1.1. Humor. It is a commonly known fact that the phenomenon of humor is an important part of our lives. Fortunately, we do not have to rely solely on our common sense to appreciate the benefits of humor, as there are numerous works that have shown its necessity in the world of humans. To name only few, we have the research conducted by Mobbs, showing that humorous stimuli activate the same areas of the human brain as such pleasant actions like listening to good music, eating tasty food, or even having sex [1]. This gives us two important clues: first, that humor is generally a positive phenomenon in our lives and second, that it can activate ("wake") the brain. The latter function is especially useful in applications such as car navigators (see Section 1.3).

Also, there is the research of Vilaythong et al., showing that exposing people to humorous contents (like funny videos) increases their feelings of hopefulness and makes them feel better [2]. Other studies showed that humor can effectively relieve our stress [3], help in dealing with anxiety [4], or even increase pain tolerance [5]. These features were used also in medicine, in so-called "laughter therapy," which

was shown to be successful in dealing with even such heavy diseases as ankylosing spondylitis (chronic inflammatory spine arthritis, also known as Bechterew's disease [6]).

Humor was also shown to work well in our interaction with other people. According to Cook and Rice [7], a sense of humor in another person actually increases the perceived benefits of a relationship. Sprecher and Regan [8] proved that a sense of humor is one of the main characteristics we use when choosing a partner. Finally, we have the research of Mulkay [9], showing that we tend to joke when we discuss difficult subjects with others. This leads to the conclusion that humor actually makes our conversations easier. These findings are consistent with the results of our previous experiments (self-reference removed due to the blind review process), in which we showed that adding a humor generator (even a simple one) to a chatterbot can significantly improve its performance and quality in the eyes of users.

1.2. Processing Funniness—Existing Work and Applications. Although, as mentioned above, humor processing is still far from thriving, in recent years some work has been done in this field. The findings of such research as those mentioned in Section 1.1 are encouraging enough to try using them also in computer science. Most of the projects by far are more or less strongly related to NLP, as the linguistic aspect of humor is relatively more computable than others. In fact, there is even a well-defined genre of jokes, called "puns" or "word plays" (also "linguistic humor"), which is based on such ambiguity features of the language as homophony or polysemy. A good example of a pun is the well-known "deer joke":

> How do you call a deer with no eyes?
> No-eye deer,

in which the funniness comes from the phonetic similarity between two phrases: "no idea" and "no-eye deer."

As they are based on the features of the language itself, puns are especially popular in the field of computational humor. Most of research by far have been focused on this genre and tried to construct pun generating engines. In this section we focus on such linguistic humor generators, paying particular attention to their practical applications, especially those that include interactions with users. Unfortunately, not many successful works exist that would incorporate pun generation into a normal human-computer dialogue.

Probably the best known and most important contribution in the field of humor processing was made by Binsted, in her research on a punning riddles generator for English [10] and Japanese [11]. The generator, named JAPE (Japanese version: BOKE) using a set of symbolic rules and a large natural language lexicon, was able to produce question-answer puns, such as

> How do you call a murderer with fibers?
> A cereal killer.

Some of the jokes generated by JAPE were judged (by schoolchildren) to be as funny as similar jokes generated by humans.

Although the results of JAPE's experiment were quite impressive, the applicability of the system is quite questionable. The number of situations in which, in a daily life, an average user would like to use a system that only generates simple punning riddles, is by all means limited. In other words, the generator of riddles may work pretty well, but it may not be extremely useful as an application.

One attempt of implementing the JAPE generator into the dialogue system Elmo was made by Loehr. It ended with the conclusion that it is difficult to arrange a generation of jokes that would be relevant to what users say and that the genre of punning riddles is generally difficult to use in a dialogue [12].

Another attempt at creating a practical application with the use of JAPE was made by Ritchie et al. [13]. The algorithm of JAPE was improved in several ways, and an interactive user interface was added in order to make interaction with humans possible. The target users of the system were children with complex communication needs (CCNs), which can result in lower levels of literacy. The software, named STANDUP, was "a language playground, with which a child can explore sounds and meanings by making up jokes, with computer assistance [13]." The results of evaluation experiments showed that the system was highly appreciated by the participants (children with CCNs) and led to improvement of their communication skills.

The application, although quite impressive, still shares JAPE's several drawbacks. Although the quality of the output was improved, the system still generates a very simple sort of punning riddles, which are not integrated into a normal, daily dialogue. Therefore, the system, albeit very useful for children with CCN, does not seem very interesting for an average, healthy user.

The JAPE system was also translated into Japanese [11]. Its output, however, was not evaluated very highly by the users, and the main problems of JAPE (simplicity and isolation of jokes) remained unsolved.

Other pun generating systems, such as McKay's WISCRAIC [14], actually shared JAPE's drawbacks—the output was very simple and it is difficult to imagine it being used in a normal interaction between humans and computers. The WISCRAIC system generates simple idiom-based 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").

This itself seems quite interesting, and the author claims that the system can be useful for nonnative English learners, as the presence of humor makes it easier to memorize words and phrases (in this case idioms). Assuming this to be true, it is still far from using humor in a natural interaction (conversation). Although WISCRAIC is said to derive information from the word context, the output itself is still a stand-alone, isolated form.

Another attempt at implementing humor into an Alice-type chatterbot was made by Augello et al. [15]. The jokes

were not generated, and the talking system would simply ask the user if he/she wants to hear a joke and what it should be about. The setup is somewhat artificial, but at least the jokes are placed in a more or less natural surrounding, namely, the dialogue.

After constructing a cross-reference joke generator, Tinholt and Nijholt [16] also tried to implement it into a chatterbot. The input for the generator was a sentence (utterance), which in itself makes it more plausible for using humor during a conversation. The following jokes generated by the system are sort of humorous misunderstandings:

> **User**: Did you know that the cops arrested the demonstrators because they were violent?
> **System**: The cops were violent? Or the demonstrators?:)

Thus, implementing the generator into a chatterbot seemed like quite a good idea. However, it turned out that this type of cross-reference ambiguity, which gives a possibility to create such misunderstanding jokes, occurs very rarely in real-life conversations. This made user-focused evaluation of the system impossible and the system was evaluated by having it analyze several chat transcripts and a simple story text [16].

Also, in one of our earlier works [17] we proposed a humor-equipped chatterbot, able to generate simple puns during nonconstrained conversations with users. The evaluation experiments showed that most users assessed the system with humor as generally better, more interesting, and making them feel better than a similar system without humor. Also results of an emotiveness analysis of chat logs showed that in most cases users' emotive states changed to more positive during and after conversations with the system with humor. For more details, see [17].

1.3. Possible Real-Life Applications. One may wonder if the whole research on joke generators is really that important. In the end, listening to jokes is fun, but how often would an average user want to do that?

Indeed, we also face these questions. In fact, this is why we are so concerned with implementing humor generators into chatterbots or systems that interact with users. Putting jokes into a conversation can make it better, more interesting, natural, and easier to conduct (see Section 1.1), which we also showed in our previous research (self-reference deleted).

Although, as mentioned in Section 1.2, there are some existing research projects on computational humor, the field is still heavily neglected, as investigating humor as a phenomenon does not sound very scientific to many researchers. This cannot be said about the world of business, though, as big companies have started to appreciate the role of humor in the workplace. Corporations such as IBM and Kodak some time ago started to employ "humor specialists"—people who are supposed to stimulate employee creativity and improve their performance with humor [18].

Also machines generating humor are in the eye of interest of, for example, car making companies, working on intelligent car navigators which are able to entertain drivers by talking and, possibly, by joking (as mentioned in Section 1.1, humorous stimuli activate the human brain). In fact, our research presented in this paper was also partially financed by one such company (acknowledgement deleted due to blind review).

1.4. Japanese Homophones. In Section 1.2 we stated that puns are one of the most computable humor genres, as they use ambiguity features of the language itself as the source of funniness. One such ambiguity phenomena is homophony, present in virtually every existing language. Some languages, however, are extremely rich in homophonic phrases. Due to its syllabic structure, one such language is Japanese, which makes it a perfect subject of research on computable humor.

1.5. Our Contribution. As mentioned in Section 1.2, existing research projects on humor generators face some serious problems, one of which is the lack of natural interaction with surrounding, leading to low usability of the system. In our research, we are trying to solve this problem. In our previous work, we described an algorithm that—based on Dybala's complex classification of Japanese puns [19]—generates puns in Japanese (self-reference deleted), and added it into Higuchi's et al. chatterbot [20]. The system generated joke-including answers using the interlocutors' utterances as an input, in order to make them at least partially relevant to what the users say. Below we present an example of the system in action:

> **User**: *Kaeru daikirai!* (I hate frogs!)
> **System**: *Kaeru to ieba tsukaeru no desu ne.* (Speaking of frogs, we could use that!).

Evaluation experiments showed that the humor-equipped chatterbot was actually appreciated and found to be better than a nonhumor-equipped one by users (self-reference deleted). Unlike other existing pun generators (where puns were generated as isolated forms), ours was successfully integrated into a normal conversation and worked robustly during evaluation experiments, in which the users were actually talking with the system.

The pun generator we used to construct the humor-equipped chatterbot (see Figure 1) was quite simple, as it generated only puns that were based on existing words (not whole phrases). In this paper we present a more complex method of generating pun candidates, using the Internet instead of hand-made databases and lexicons. We also introduce a novel method of filtrating the candidates list (which is often quite long) in search of the most plausible phrases that can then be integrated into pun-including sentences.

1.6. The Research Questions. The questions we aim to answer in this work are as follows.

(i) Is the constructed algorithm able to generate pun candidates usable in pun generation?

(ii) Is there any method to reduce the amount of pun candidates using the Internet as a linguistic source?

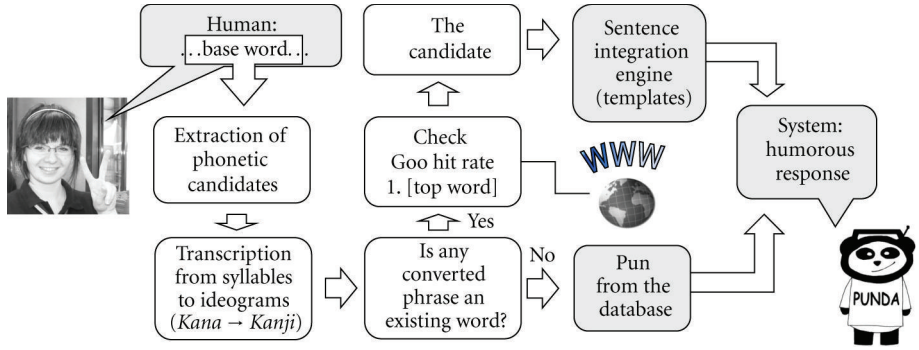


Figure 1: Pundalin (humor-equipped chatterbot) pun generation algorithm outline.

(iii) Are there any regularities in positions of base phrases and/or candidates in any of conducted web rankings?

(iv) Are there any correlations between any of conducted web rankings?

(v) Can any of these information be used to reduce the amount of pun candidates?

The answers to these questions are summarized in Table 3 (see Section 4).

2. Pun Generation Algorithm

The algorithm presented in this section generates pun candidates using not only single words, but also homophonic (or phonetically similar) phrases. In the evaluation experiment (see Section 3) we tested the system's performance on human-created puns. The results showed that the system's accuracy was on the level of 72.5%, which is promising enough to continue the research.

Another problem, often mentioned by humor researchers (as, for example, [21]), concerns the quantity of possible variations when generating similar sounding phrases. In some cases, the possibilities of phonetic transformation are very complex, especially if the query is not only limited to single words, but also includes phrases. In our research we also faced this problem. For example, for the word 遺体 (*itai*—a corpse), our system—in its minimal settings—generates above 100 candidates. Although we are happy to see the algorithm working that prolifically, choosing a proper candidate from such a long list is quite a challenging task. Thus, in this paper we introduce a novel method, which allows us to limit the number of pun candidates generated by the system (see Section 4). Having this implemented, we are only one step (sentence integration) away from constructing a fully functional, complex phrase pun generating engine.

2.1. Pun Generation Patterns. In our research, we base our work on a complex Japanese pun classification proposed by Dybala [19]. The puns were divided into 12 groups (with numerous subgroups), according to mora (~syllable) changes between the base phrases and phrases transformed into a pun. For example, in a simple pun "*kono kusa wa kusai*" ("this grass stinks"), the base phrase "*kusa*" (grass) is transformed into "*kusai*" (to stink), and the technique used is called "final mora addition", as there is one mora ("*i*") added to the end of base phrase.

Figure 2 presents the outline of Dybala's classification with pun examples and mora listed changes.

The classification was used in our research to create pun generation patterns (an equivalent of JAPE's schemata). For example, the group "final mora addition" (see Figure 2) gives us the pattern that can be transcribed as (base phrase)(*), where (*) means one single mora. Currently, there are seven patterns implemented in the system (presented below, with the word *katana* as an example):

(1) homophony ([*katana*]),

(2) initial mora addition (**katana*: *akatana, ikatana, ukatana...*),

(3) internal mora addition (*ka*tana, kata*na: kataana, kataina, katauna...*),

(4) final mora addition (*katana*: katanaa, katanai, katanau...*),

(5) final mora omission (*kata*),

(6) internal mora omission (*kana*),

(7) mora transformation (*gatana, tatana, matana...*).

In pattern 7 the number of possible transformations is very large (assuming that any sound can be transformed into any other sound), therefore, in our system we used Japanese phoneme similarity values, proposed by Takizawa et al. [22] to identify phrases that sound more similar than others.

The patterns were used in the pun candidates generation algorithm, described in the next section.

2.2. Pun Candidates Generation. Having extracted pun generation patterns (see Section 2.1), we used them in the pun candidates generation algorithm (see Figure 3 for the outline). To choose the right phrases from the phonetic candidates list, we decided to use the Internet instead of hand-made lexicons and dictionaries (see Section 2.2.1).

2.2.1. Internet as a Source of Information. Other existing pun generators (e.g., JAPE or BOKE) used hand-made lexical resources or general-purpose lexicons, such as WordNet [23].

(1) Homophony
(カエルが帰る*Kaeru ga kaeru* <The frog comes back>)
(2) Mora addition
2.1 **Initial mora**
(スイカは安いか *Suika wa yasuika* <Watermelon is cheap>)
2.2 **Final mora** (カバのかばん *Kaba no kaban* <Hippo's bag>)
2.3 **Internal mora**
(布団が吹っ飛んだ *Futon ga futtonda* <Futon flew away>)
(3) Mora omission
3.1 **Final mora**
(スキーが好き *Sukii ga suki* <I like skiing>)
3.2 **Internal mora**
(ステーキはすてき *Suteeki wa suteki* <Steaks are cool>
(4) Mora transformation
4.1 **Consonant transformation**
(目だまし時計 *Medamasidokei* <Eye-misleading clock>)
4.2 **Vowel transformation**
(背中の悲劇 *Senaka no higeki* <Senaka's tragedy>, senaka = back)
(5) Phoneme and syllable metathesis
(漫画を読むのはま、我慢 *Manga wo yomu no wa ma, gaman* <I can stand reading commics>)
(6) *Kanji* reading change
(食王 *Syokkingu* <Shocking>)
(7) Morpheme metathesis
(男を売る思い出*Otoko wo uru omoide* <A memory of selling a man> - From the title of tv drama「思い出を売る男」*Omoide wo uru otoko* <A man selling memories>)
(8) Blend
(老いてはことをし損ずる *Oite wa koo wo sisonzuru* <When you get old, you start to make waste> - A blend of two proverbs:「急いてはことを仕損ずる」*Seite wa koto wo sisonzuru* <Haste makes waste> and「老いては子に従え」*Oite wa ko ni sitagae* <When you get old, you should listen to your children>)
(9) Division
(ゆで卵をゆでたのは孫 *Yudetamago wo yudeta no wa mago* <It's the grandchild who boiled the egg>)
(10) Quiz
(はなしの話は？なし！ *Hanasi no hanasi wa? Nasi!* <How would you call a talk without teeth? A pear.>)
(11) Mix of Japanese and foreign languages
(戸部君、ハムレットが君に言ってるだろう、トベ・オル・ノットベ *Tobe-kun, Hamuretto ga kimi ni itte iru darou, tobe oru nottobe* <Tobe, Hamlet is talking to you – To be or not to be>)
(12) Pause transference
(金をくれ、頼む！・金をくれた、飲む！ *Kane wo kure, tanomu! – Kane wo kureta, nomu!* <Please, give me some money! –You gave me the money, let's drink!>)

Figure 2: Dybala's Japanese pun classification (with examples).

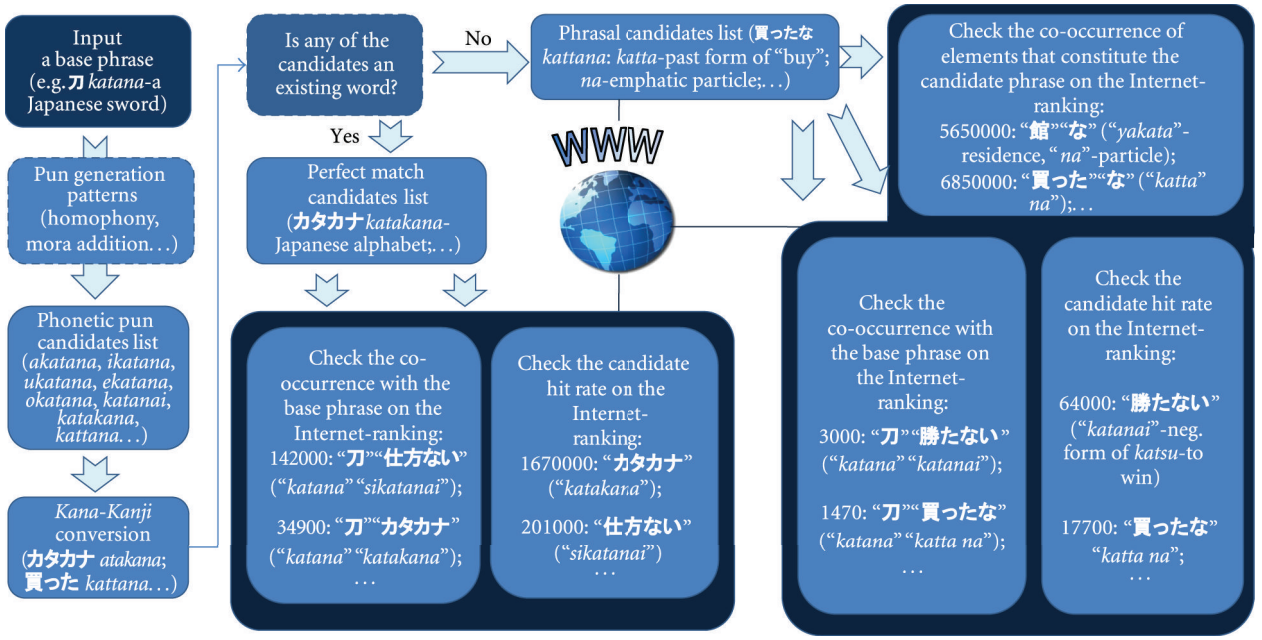


Figure 3: Outline of pun candidates generation algorithm.

Creation of the former is very laborious, and the latter is not always adaptable to the purpose of joke generation. Both these types of resources share the same drawback, namely, they are constant, that is, do not follow the changes in language and thus are not up to date.

Contrary to these, the Internet is a dynamically evolving, easily accessible, large-scale database, and full of information, not only about phonetics or semantics, but also about the world in general. Even in this very moment, millions of users are writing their blogs, entries in Wikipedia and articles on every possible topic. The search engines, allowing us to find those parts of that ocean of information we need, are becoming faster and more sophisticated. This makes the Internet a well-suited resource for our pun generation engine.

In our algorithm, we use the Internet to construct three types of pun candidates rankings:

(1) co-occurrence of the elements that form the candidate phrase

(2) candidate phrase hit rate ranking

(3) co-occurrence of the candidate phrase and the base word.

The experiment (see Sections 3 and 4) showed that (2) and (3) are similar enough to be used to filter the candidates list.

2.2.2. The Algorithm. The outline of our pun generation algorithm is presented in Figure 3.

The input is a word or a phrase, which becomes the base phrase for a pun. When (as in our previous experiments) the system is implemented into a chatterbot, the base phrase is extracted from the user's utterance. Preliminary analysis of human-created jokes showed that in most cases the base phrase is a noun or a noun phrase. In this work we used this assumption and set up the system to first look for nouns. If there are any, it queries the sentence for adjectives, verbs and then for any other phrases (in this order).

Next, the base phrase is transformed using the pun generation patterns (see Section 2.1), this is how the phonetic candidates list is generated. To this point, everything happens in *Hiragana* Japanese syllable alphabet. In the next step, each of the phonetic candidates is converted into *Kanji* (Chinese characters), using MeCab-skkserv *Kana-Kanji* Converter (MeCab-skkserv *Kanji-Kana* converter, http://chasen.org/~taku/software/mecab-skkserv/). As in many cases, not only one conversion is possible, we decided to generate only two different options whenever it was possible. Any further increase of possible transcriptions would give us an enormous amount of queries to be made, which, in turn, would be very time consuming.

In the next step, all of the converted candidates are analyzed by the POS and morphological analyzer MeCab (Kudo, T., MeCab: Yet another part-of-speech and morphological analyzer, http://mecab.sourceforge.net/ (2001)) to check if any of them is an existing word. Those which are found to be, form a list of perfect match candidates and the rest form a list of phrase candidates.

Next, each entry on the list is checked on the Internet (current version of the system uses the Yahoo search engine (http://www.yahoo.co.jp)). For the phrase candidates list, the system performs three types of queries: (1) co-occurrence of the elements that form the phrase check; (2) the whole phrase hit rate check (as an exact match); (3) the co-occurrence of the whole candidate phrase and the base phrase check. For example, if the base phrase is *katana* (a Japanese sword) and one of the candidates is *katta na* (*katta*—past form of *kau*—to buy; *na*—emphatic particle), the three queries will look like this: (1) "*katta* " "*na*"; (2) "*kattana*"; (3) "*katana*" "*kattana*."

For candidates that are actually existing words, checking of co-occurrence of elements that form the phrase is in fact identical with the whole phrase check. Therefore, for the perfect match candidates list, the system performs only two types of queries: (1) the whole phrase hit rate check and (2) the co-occurrence of the whole candidate phrase and the base phrase check.

Having finished the queries, the system forms the ranks for each method, sorted from the highest to the lowest hit rate. Figure 5 shows the results for the word 買い物 (*kaimono*: shopping).

3. The Experiment

To verify the algorithm and check if the candidates generated by the system are usable, we conducted an evaluation experiment. From Sjöbergh's Japanese human-created pun database [24] we chose 200 jokes and used their base phrases as our system's input, making it generate as many candidates as possible. Then, we checked if the phrase that was actually used in the joke was among the candidates generated by the system. For example, from the joke: *Katana wo katta na* (bought a Japanese sword, you know) we extracted the base phrase *katana* (a Japanese sword) and used it as an input for our system. If somewhere on the final candidates list we found the phrase *katta na* (bought), it meant that the system actually generated the right candidate.

In our previous research (on humor-equipped chatterbots) we used only a simplified version of the algorithm, generating only exact match candidates. Thus, in this experiment we decided to compare two sets of results: one similar to the previous experiments (exact matching candidates only) and one with the phrase candidates generation partly added.

As shown in Figure 4, when only exact match candidates generation was used, the system was successful only in 43.5% of cases, which is not a very impressing result. However, after adding the phrase candidates generation, the accuracy was visibly improved and reached 72.5%. We believe that this result, albeit still far from being perfect, is promising enough to continue our research.

As mentioned above, in the step of *Kanji* conversion, each phrase was converted in two possible ways. In some cases there were more possibilities of conversion, however, further increase of the candidate amount would extend the time needed to perform the generation. Thus, in the experiment for some puns, proper candidates were not generated, although they would be, if the amount of possible transcription was increased. The maximum number of possible conversions allowed by the MeCab-skkserv *Kana-Kanji* Converter is 100, which is too many for our purpose—however, some minor increase (up to 5 or 6) should slightly improve the system's results. For example, from the joke "*Gokai ni iku to gokai sareru*" (when you go to the 5th floor, it is a misunderstanding), we extracted the base phrase 五階 ("*gokai*": 5th floor), which was then transformed into pun candidates. The algorithm, trying to generate homophonic phrases, generated two possible transcriptions: 5回 ("*gokai*") and 五回 ("*gokai*": both phrases meaning "five times"). If we set the MeCab-skkserv *Kana-Kanji* Converter to generate five possible transcriptions, the word 誤解 ("*gokai*": misunderstanding) would be among the candidates. This, however, would largely increase the general amount of propositions and Web queries, which in turn would lead to the extension of time needed to generate pun candidates. As we use an Internet search engine (currently Yahoo), the amount and frequency of allowed queries are restricted and if the system puts too much load on the search engine, it can be recognized as a spammer and temporarily blocked. Therefore, we had to put pauses between each query, which caused quite a severe extension of the time needed to generate the candidates. The

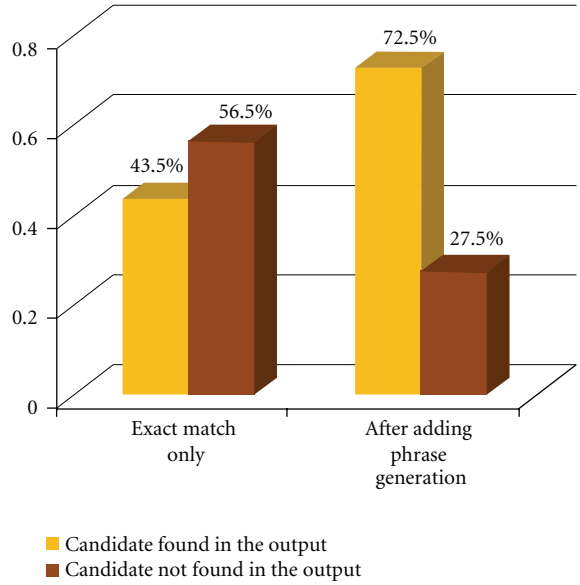


Figure 4: Evaluation experiment results—before and after adding phrase generation algorithm.

time differs depending on the amount of generated phonetic candidates—in total, it took the system about one week to analyze pun candidates for the 200 base phrases used in the experiment. This in itself is quite long, and a further increase of the number of queries would increase the time even more.

4. The Number of Candidates

The results of the evaluation experiment (see Section 3) were satisfying enough to move to the next step of our research, which is the choice of the right candidate and its integration into a sentence form. However, by looking at the system's outputs, we realized that at this point we face one serious problem: the amount of generated pun candidates is generally quite vast, in some cases exceeding even 100 propositions. On one hand, this is an indication that the system works well and is very prolific, but, on the other hand, it is quite difficult to choose which phrase should be selected to form a pun.

Therefore, we decided to look for a method that would allow us to reduce the number of candidates (if not choose the right one).

To do this, we analyzed the results of the three web-queries-based rankings (see Section 2.2). Our first assumption was that at least in one of them there should be a visible tendency, showing that, for example, candidates that were actually used in the human-created puns were at the top or relatively near the top of the ranking. This, however, turned out not to be true. Checking the positions of phrases actually used in the human-created jokes showed that their distribution was quite random—varying from the top or near the top to the very bottom of the ranking. Thus, it was necessary to search for some new solutions.

Table 1: Average position of the words from human-created puns in three rankings.

Ranking	1. Co-occurrence of elements that form the candidate	2. Candidate phrase hit rate	3. Candidate co-occurrence with the base phrase
Average position	9.7	5.5	5.2

Next, we decided to compare all of the rankings and positions of candidates. Figure 5 shows the results of such comparison for the base phrase 買い物 (*kaimono*-shopping), taken from the joke 買い物は若いものに頼もう (*kaimono wa wakai mono ni tanomou*—let us ask the young people to do the shopping).

As shown in Figure 5, in the whole phrase hit rate ranking and the co-occurrence of the whole candidate phrase and the base phrase ranking, the word 若いもの (*wakai mono*) that was actually used in the human-created pun is on the same position. This, in fact, turned out to be an overall tendency in the results—the average position of phrases used in human-created jokes in these two rankings is generally very similar (see Table 1), and the average difference between them is only 0.3. Contrary to that, in the case of ranking 1, the average position of phrases from human-created jokes is 9.7, which is significantly different from the remaining two.

The similarity of rankings 2 and 3 opened some new possibilities for our research. In the next step, we decided to check in how many cases the human-created pun phrase is on the same position in both of the rankings.

The results were not that great—the position was exactly the same in 46.2% of cases. Then, we tried to slightly extend the scope of the search and checked in how many cases the positions of phrases from human-created jokes differed by 1.

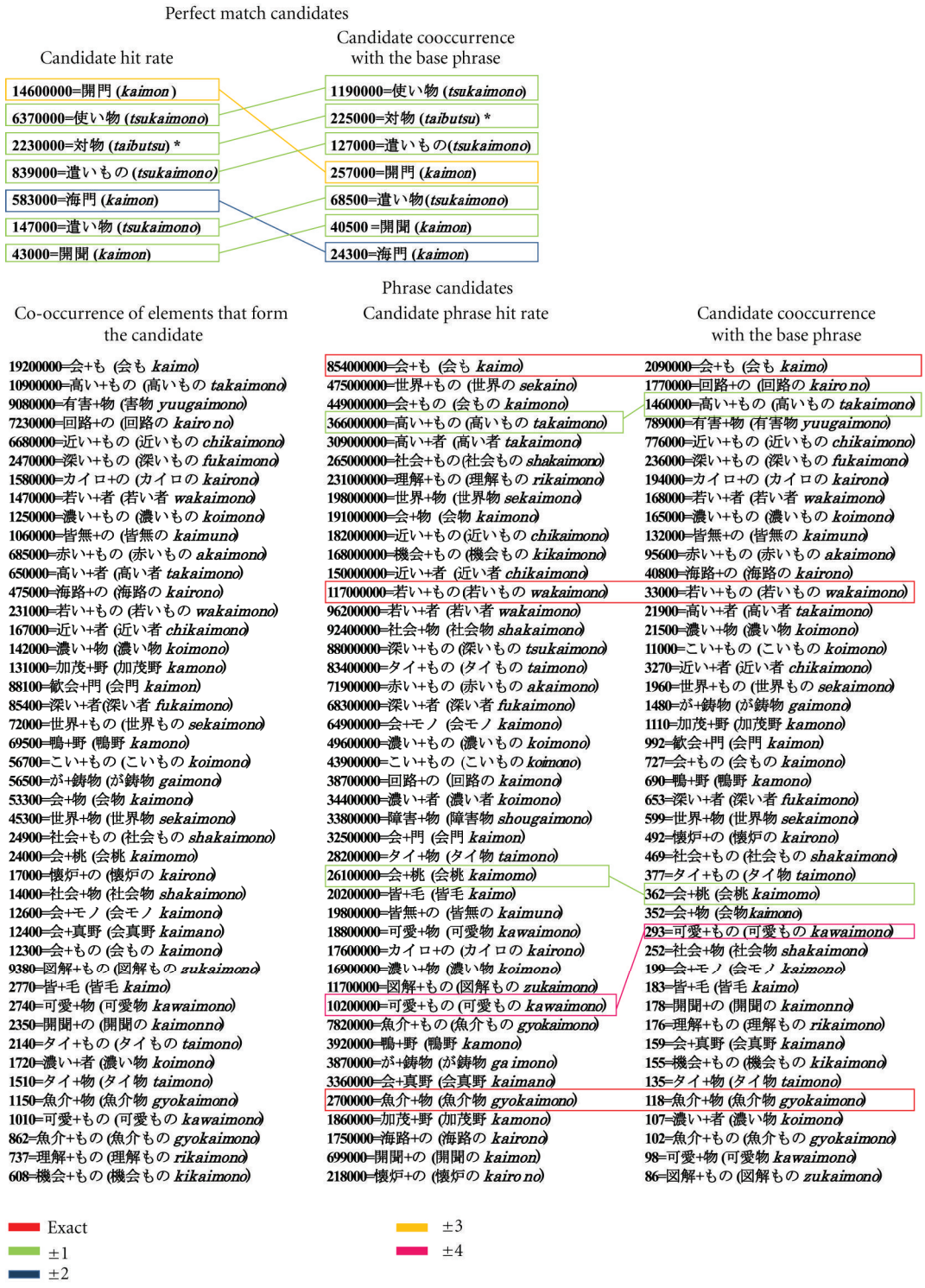


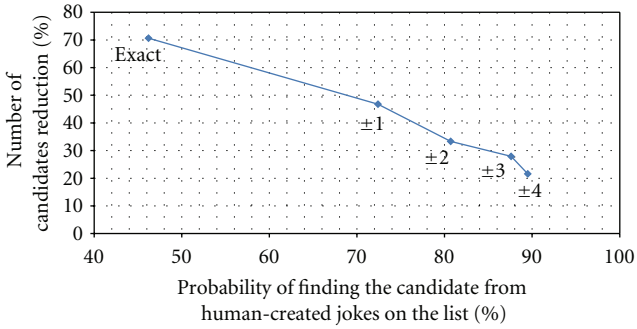
Figure 5: Output example for the word 買い物 (*kaimono*: shopping).

Here the results were visibly better, as in 72.4% of cases the correct phrases could be found that way. Next, in a similar way, we checked how the results change when the positions differed by 2, 3, and 4. The results are summarized in Table 2 and Figures 6, 7, and 8.

As depicted above, the average candidates reduction is very high (70.5%) if we take into consideration only these

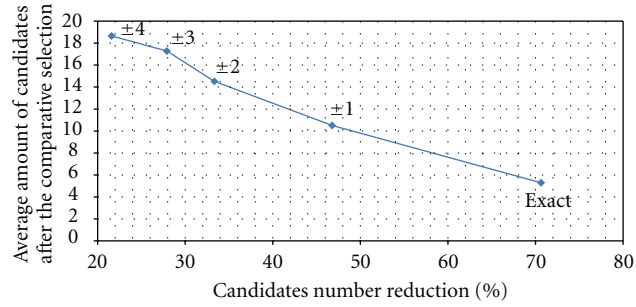
Table 2: Correlation between the average number of candidates after comparative reduction, probability of finding the candidate from human-created pun on the list after reduction, and average percentage of candidates amount reduction.

	Exact	±1	±2	±3	±4
Average number of candidates after reduction	5.30	10.51	14.52	17.28	18.65
Probability of finding the candidate from human-created pun on the list after reduction	46.2%	72.4%	80.7%	87.6%	89.5%
Average percentage of candidates amount reduction	70.6%	46.8%	33.3%	27.9%	21.6%



Exact: when positions in both rankings are exactly the same
±1: when positions in both rankings differ by 1
±2: when positions in both rankings differ by 2
±3: when positions in both rankings differ by 3
±4: when positions in both rankings differ by 4

Figure 6: Correlation between the average reduction of candidates number and probability of finding the candidate from the human-created pun on the list after reduction (%).



Exact: when positions in both rankings are exactly the same
±1: when positions in both rankings differ by 1
±2: when positions in both rankings differ by 2
±3: when positions in both rankings differ by 3
±4: when positions in both rankings differ by 4

Figure 7: Correlation between average number of candidates after the comparative selection and average percentage of candidates amount reduction.

candidates that are on exactly the same places in both hit rate and co-occurrence with base phrase rankings. In such case, however, only in 46.2% of cases was the phrase that was actually used in the human-created joke found. Therefore, if we are ready to sacrifice a small part of the reduction percentage, we can raise the probability of finding the right phrase on the list by taking into consideration candidates whose positions differed by 1 in both rankings. The reduction rate falls down to 46.2% (so it still removes half of the candidates), and the chances for finding the right word are visibly higher. However, if we still want to raise them at the cost of reduction percentage, we can check the candidates whose positions in the ranking differ by 2 (33.3% of reduction, 80.7% of probability), by 3 (27.9% of reduction, 87.6% of probability), or by 4 (21.6% of reduction, 89.6% of probability).

As for the example presented in Figure 5, the reduction here was fairly successful, as the number of candidates decreased by 94.1% for an exact match, by 80.4%% when the positions differed by 1, by 78.4%% when they differed by 2, 76.5% when differed by 3, and 74.5% when differed by 4. The word that was actually used in the human-created pun 若いもの (*wakaimono*: young people) was found on the exact match list. Also other candidates that remained after the reduction procedure seem usable, for example, the phrase 高いもの (*takaimono*: something high and expensive) could be used to create a meaningful pun, as it is semantically related to the base phrase 買い物 (*kaimono*: shopping).

This, however, does not mean that all of the candidates after the reduction are perfectly usable. One of the perfect match (word) candidates–対物 (*taibutsu*: pertaining to objects, marked with an asterisk * in Figure 5) is an effect of a wrong *Kana-Kanji* transcription by MeCab-skkserv (the 物 character can be read *butsu*, and also *mono*). Therefore, the phonetic similarity here is very low and there is no possibility of creating a pun. An additional algorithm to fix such mistakes is needed.

5. Discussion

The results presented above show that there actually are some similarities between the two rankings (candidate phrase hit rate and co-occurrence with the base phrase), which can be used to reduce the amount of generated candidates. The reduction affects the system's accuracy—the more candidates we delete from the list, the less chance we have of finding the right one among those that survived the selection. Therefore,

Table 3: Results for research questions.

Question	Answer
Is the constructed algorithm able to generate pun candidates usable in pun generation?	YES The algorithm generated pun candidates with accuracy of 72.5%.
Is there any method to reduce the amount of pun candidates using the Internet as a linguistic source?	YES Conducting web frequency rankings is sufficient to significantly reduce the amount of candidates.
Are there any regularities in positions of base phrases and/or candidates in any of conducted web rankings?	NO Position itself is not sufficient to state whether a candidate is appropriate to be used in a pun or should be reduced.
Are there any correlations between any of conducted web rankings?	YES Among three web frequency rankings conducted, the correlation between the candidate phrase hit rate and the candidate co-occurrence with the base phrase
Can any of these information be used to reduce the amount of pun candidates?	YES The amount of candidates can be successfully reduced using the correlation mentioned above.

instead of setting a rigid rule (for instance, that only the candidates which positions in both rankings vary by 2 should be taken into consideration), we decided to describe the whole set of results here—just to give us an opportunity to flexibly adjust the method to the needs of any particular case.

Some implications of the results are discussed below.

5.1. Is There Really One and Only Right Candidate? In the evaluation experiment (described in Section 3) we assumed that the system works properly if it generates candidates that were actually used in human-created jokes. In other words, the assumption was that the phrase chosen by humans from the set of possibilities is by all means the right one. This, however, does not necessarily have to be true, as we can imagine a situation, in which the system would choose another candidate than a human would and generat a joke which would be evaluated as more funny than one created by a human. For example, for the base word 白菜 (*hakusai*: Chinese cabbage, see Figure 2), the system generated the phrase は臭い (*ha kusai*: stinks), which was actually used in the human created joke; however, on the candidates list there was also the phrase は臭い (*ha kusai*: teeth stink), which also seems usable in a pun generation. Also other candidates, if properly integrated into a sentence, seem at least not completely useless in terms of pun generation.

For these reasons, rather than thinking of a method that would eventually allow us to reduce the candidates list only to the one that was actually used in a human-created pun, we decided to let the system generate a whole list of candidates. As in many cases the list is too long and includes candidates that are possibly less usable (due to, e.g., their rareness), we proposed an algorithm for reducing the amount of candidates. This method does not allow us to select one final candidate, but reduces their number to those, which, in most cases, are usable to make a pun.

5.2. Assessing Usability. As far as the usability of pun candidates is concerned, its evaluation is generally quite a troublesome issue. Having completed the algorithm of candidates generation, we intended to conduct an experiment in which human evaluators would assess if every particular candidate is usable (possible to be used in a pun). However, it occurred to us, that this approach is not necessarily proper—even if the evaluator says that the particular phrase cannot be used to create a joke, the system could potentially do that and actually generate a pun. Therefore, we decided to abandon this idea at this stage of system development and conduct human involving evaluation experiments after finishing the pun generation algorithm. Instead, we are planning to focus on the process of further filtration of the candidates and methods of selecting the best and most usable candidate (see Section 5.4).

5.3. Rankings and What Next? In Figures 6, 7, and 8 and in Table 2 we presented the correlations between the average amount of candidates after comparative reduction, the probability of finding the candidate from human-created pun on the list after reduction, and the average percentage of candidates amount reduction. We checked the results for 5 cases: when the candidates position in both rankings are exactly the same, when they differ by 1, 2, 3, and 4. This in itself is quite interesting, but the question is how can we use these results?

The answer is it all depends on the priorities. In our research, we are using Internet to generate the candidates and will be using it even more in the next step (plausible candidate selection and sentence integration module). The web mining process is very time consuming, as to avoid being blocked by search engines we needed to deliberately delay the queries. As we are implementing our joke generators into a chatterbot, which interacts with users in the real time, every delay in generating the system output makes the interaction go less smoothly. Therefore, in this project the priority for us is the reduction of the number of candidates. We do not, however, want to loose too many potentially good candidates just because the system could not handle a few more queries—therefore, for the needs of this particular project, looking at " ± 1" candidates seems quite optimal. We reduce the candidate amount by almost half (in the experiment data 46.8%) and the probability of finding a good candidate is reasonably high (in the experiment data 72.4%). Needless to say, the percentage may differ for every particular input, and the data given above describe rather tendencies

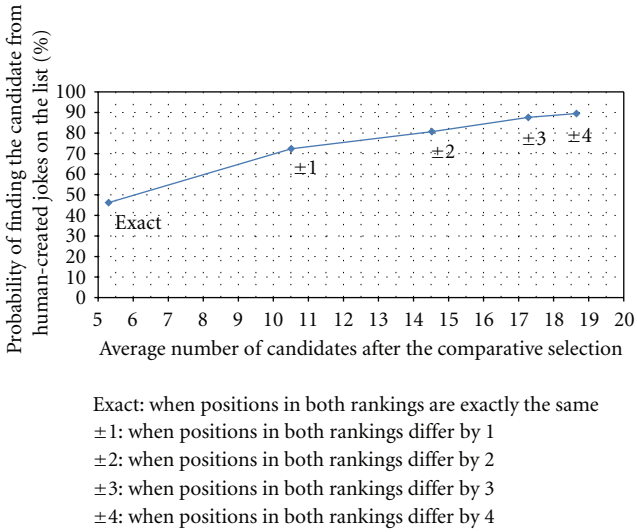


Figure 8: Correlation between the average number of candidates after the comparative selection and probability of finding the candidate from the human-created pun on the list after reduction.

and chances. However, as such, they are quite usable when deciding which method to chose and which priorities should be made in the process of pun generation.

So, provided that the reduction of the number of candidates, combined with reasonably good chances of still finding proper words, is our priority, should we decide that we only take the "±1" candidates into consideration? It does not have to be that simple—maybe we should set a sort of threshold, a minimal amount of candidates we want to have on the list? If, for example, we want to have at least ten positions on the output list, the system could continue checking the rankings until it finds the desired amount (i.e., first it would check the candidates with exactly the same position, then, if the number is still below the threshold, continue checking ±1, ±2, ±3, and ±4 candidates).

Both of these methods seem quite interesting and worth further investigation. Nevertheless, the main issue here is still the processing time problem. Currently we are conducting experiment with an offline text corpus, which shall result in significantly shortening the process.

5.4. How to Choose the Right Candidate? There are (at least) two possible methods to do this. The first is based on an assumption that—after the list reduction—every candidate can be used in a pun, as long as it can be properly integrated into a sentence form. This idea is quite simple, but may be worth trying—after all it can be argued, that it is the sound (phonetics) that matters the most in puns, and as long as the candidate is phonetically similar and somewhat (after checking the co-occurrence) related to the base phrase, it may be enough to form an output that would be recognized as a pun.

We can, however, choose another solution and try to use some existing online association engine, as for example Reflexa (http://labs.preferred.jp/reflexa/) or one used by Higuchi et al. [20], in order to find out how each of the candidates is related to the base phrase. We do not claim that the most related word would be the best choice—we may, however, find some regularities and tendencies here (as we did in the ranking similarities described above), which, in turn, will allow us to improve the pun generation algorithm.

Another important issue to be taken into consideration in the pun generation process is the context of the dialogue. Words or phrases describing the topic of conversation should also be used in the association algorithm in order to determine how each pun candidate is related to the dialogue context.

To summarize, in the next step of the pun generation algorithm development we are planning to take the following into consideration:

(i) base phrases,

(ii) pun candidates,

(iii) topic and context-related words and phrases,

and examine how they are related to each other. This will be our main research goal in the near future.

5.5. Timing of Humor. Another important issue when implementing the humor generator into chatterbots is the timing of humor, that is, whether jokes are told in appropriate moments of conversations. In our earlier works (e.g., Dybala [17]) we presented an emotiveness-analysis-based timing procedure, in which an emotion-from-text detector was combined with the humorous chatterbot in order to detect users emotions during conversations and on this basis decide whether a joke should be told or not. In the primary setup, the system would tell jokes when users' emotional state was assessed as negative or neutral. Evaluation experiments showed that in most cases there was an increase in users positive emotions and decrease in negative emotions during and after the interactions. Also, in a questionnaire conducted afterwards, most users stated that the system used humor in appropriate moments (for more details, see [17]).

Although this setup still requires improvements (i.e., which specific emotions can be followed by humor), it shows that the emotiveness-based timing rule might be a step in the right direction.

5.6. How to Choose the Right Base Phrase? As mentioned above (see Section 2.2.2), the base phrase in most cases was nouns or noun phrases. However, this is not enough when implementing pun generator into a chatterbot, as another crucial criterion here would be relevance to the conversation topic in general. In order to investigate regularities in this issue, an analysis of human-human-pun-including conversations is required. However, this is quite problematic, as, to our knowledge, no corpus of such conversations exists for Japanese, which means that we need to acquire such data by ourselves. This is quite time and effort consuming, yet still remains a part of our future work.

5.7. Applicability for Other Languages. Although the system we developed generates only Japanese puns, it does not

use any pun-generation-oriented hand-made lexicons or resources, as the Internet is language independent. Also the pun generation patterns mentioned here can be found in other languages (in various configurations, depending on the language). In Dybala, [19], a comparative analysis of Polish and Japanese linguistic jokes was conducted, showing that most generation patterns are in fact present in both of these languages.

Therefore, in order to translate the system into other languages one would have to decide which generation patterns are present in the target language, and then find a set of appropriate NLP tools. As this does not seem excessively laborious, in the near future we are planning to create an English version of our generator.

6. Conclusions

As described above, we have successfully constructed a phrasal pun candidates generation module. The engine is much more sophisticated than the one we used in our previous research on joking chatterbots (see Figure 1). Thus, having finished the candidates generation part, the next step will be to implement it into the chatterbot.

The system described here generates phrasal pun candidates (one step before actual phrasal pun generation). However, in its current shape it can be used as a pun generation support tool—if, for instance, a user would like to make up a pun from a particular word or phrase, the system's output (candidates list) can be quite helpful here. This could be used for a purpose such as generating funny commercial slogans.

Summary

The results of the work presented in this paper are summarized in Table 3 in the form of answers to the research questions stated in Section 1.6.

References

[1] D. Mobbs, M. D. Greicius, E. Abdel-Azim, V. Menon, and A. L. Reiss, "Humor modulates the mesolimbic reward centers," *Neuron*, vol. 40, no. 5, pp. 1041–1048, 2003.

[2] A. P. Vilaythong, R. C. Arnau, D. H. Rosen, and N. Mascaro, "Humor and hope: can humor increase hope?" *Humor*, vol. 16, no. 1, pp. 79–89, 2003.

[3] A. Cann, K. Holt, and L. G. Calhoun, "The roles of humor and sense of humor in responses to stressors," *Humor*, vol. 12, no. 2, pp. 177–193, 1999.

[4] C. C. Moran, "Short-term mood change, perceived funniness, and the effect of humor stimuli," *Behavioral Medicine*, vol. 22, no. 1, pp. 32–38, 1996.

[5] S. Svebak, K. Søndenaa, T. Hausken, O. Søreide, Å. Hammar, and A. Berstad, "The significance of personality in pain from gallbladder stones," *Scandinavian Journal of Gastroenterology*, vol. 35, no. 7, pp. 759–764, 2000.

[6] N. Cousins, *Anatomy of an Illness As Perceived by the Patient: Reflections on Healing and Regeneration*, W. W. Norton & Company, New York, NY, USA, 1979.

[7] K. S. Cook and E. Rice, "Social exchange theory," in *Handbook of Social Psychology*, J. Delamater, Ed., pp. 53–76, Plenum, New York, NY, USA, 2003.

[8] S. Sprecher and P. C. Regan, "Liking some things (in some people) more than others: partner preferences in romantic relationships and friendships," *Journal of Social and Personal Relationships*, vol. 19, no. 4, pp. 463–481, 2002.

[9] M. Mulkay, *On Humor: Its Nature and Its Place in Modern Society*, Basil Blackwell, New York, NY, USA, 1988.

[10] K. Binsted, *Machine Humour: An Implemented Model of Puns*, University of Edinburgh, UK, 1996.

[11] K. Binsted and O. Takizawa, *Computer Generation of Puns in Japanese*, Sony Computer Science Lab, Communications Research Lab, 1997.

[12] D. Loehr, "An integration of a pun generator with a natural language robot," in *Proceedings of the International Workshop on Computational Humor*, pp. 161–172, University of Twente, The Netherlands, 1996.

[13] G. Ritchie, R. Manurung, H. Pain, A. Waller, R. Black, and D. O'Mara, "A practical application of computational humour," in *Proceedings of the 4th International Joint Conference on Computational Creativity*, pp. 91–98, 2007.

[14] J. McKay, "Generation of idiom-based witticisms to aid second language learning," in *Proceedings of the April Fool's Day Workshop on Computational Humour*, Stock et al., Ed., pp. 77–87, Trento, Italy, 2002.

[15] A. Augello, G. Saccone, S. Gaglio, and G. Pilato, "Humorist bot: bringing computational humour in a chat-bot system," in *Proceedings of the 2nd International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '08)*, pp. 703–708, Barcelona, Spain, March 2008.

[16] H. W. Tinholt and A. Nijholt, "Computational humour: utilizing cross-reference ambiguity for conversational jokes," in *Proceedings of the 7th International Workshop on Fuzzy Logic and Applications (WILF '07)*, vol. 4578 of *Lecture Notes in Artificial Intelligence*, pp. 477–483, Genoa, Italy, 2007.

[17] P. Dybala, M. Ptaszynski, J. MacIejewski, M. Takahashi, R. Rzepka, and K. Araki, "Multiagent system for joke generation: humor and emotions combined in human-agent conversation," *Journal of Ambient Intelligence and Smart Environments*, vol. 2, no. 1, pp. 31–48, 2010.

[18] D. E. Gibson, "Humor consulting: laughs for power and profit in organizations," *Humor*, vol. 7, no. 4, pp. 403–428, 1994.

[19] P. Dybala, *Dajare—Nihongo ni Okeru Dōon'igi ni Motozuku Gengo Yūgi (Dajare—Japanese Puns Based on Homophony)*, Jagiellonian University Press, Kraków, Poland, 2006.

[20] S. Higuchi, R. Rzepka, and K. Araki, "A casual conversation system using modality and word associations retrieved from the web," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pp. 382–390, Honolulu, Hawaii, USA, 2008.

[21] G. Ritchie, "Current directions in computational humour," *Artificial Intelligence Review*, vol. 16, no. 2, pp. 119–135, 2001.

[22] O. Takizawa, M. Yanagida, A. Ito, and H. Isahara, "On computational processing of rhetorical expressions—puns, ironies and tautologies," in *Proceedings of the International Workshop on Computational Humor*, pp. 39–52, The Netherlands, 1996.

[23] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[24] J. Sjöbergh and K. Araki, "Robots make things funnier," in *Proceedings of the International Workshop on Laughter in Interaction and Body Movement (LIBM '08)*, pp. 46–51, 2008.