Crossing Word Borders
Towards Phrasal Pun Generation Engine

Pawel Dybala **Michal Ptaszynski** **Rafal Rzepka** **Kenji Araki**
Graduate School of Information Science
and Technology, Hokkaido University
Kita 14 Nishi 9, Kita-ku, 060-0914 Sapporo, Japan
{paweldybala,ptaszynski,kabura,araki}@media.eng.hokudai.ac.jp

Abstract

In our previous works we showed that implementing a very simple pun generator into a chatterbot can visibly improve its performance. In this paper we present a more complex pun candidate generation algorithm, that, using the Internet as a resource, is able to generate not only single words, but also phrasal candidates. The evaluation experiment showed that the system can generate pun candidates with 72.5% accuracy. We discuss the results and point out some directions for the future.

1 Introduction

In the first sections of this paper we will briefly (due to the limited lenght of this paper) summarize existing research about humor. Then, we will explain our contribution to the field of pun generating, describe the algorithm of our system, the results of evaluation experiment and some tasks for the nearest future of our project.

1.1 Humor

The importance of humor in our lives is actually not questionable, in both commonsensical and scientific ways. We all know that "a good laugh" can make us feel better and bring some refreshment to our boring reality. This benefits of humor have actually been confirmed in scientific research. To name only few of such projects – it was showed that humor can effectively relieve our stress (Cann et al., 1999) or help deal with anxiety (Moran, 1996). According to Sprecher and Regan (2002) the sense of humor is one of main characteristics we use when choosing a partner. We have also the research of Mulkay (1988), showing that we tend to joke when we discuss difficult subject with others. This leads to the conclusion that humor actually makes our conversations easier.

Above findings are consistent with what we found in our previous experiments (Dybala et al., 2008). We proved that implementing a simple humor generator into a chatterbot can significantly improve its performance and quality in the eyes of users and non-users evaluators. Thus, we believe using the humor to facilitate the HCI is a worthwhile enterprise and should be continued. This paper brings us one step further in our project.

1.2 Computational Humor

In recent years, the positive traits of humor (some of which were listed in **1.1**) started being appreciated in the world of science, also in such fields as AI or NLP. The type of jokes, that is especially popular in the latter field, is called "puns" or "linguistic jokes". The source of funniness in such jokes is grounded in the features of language itself. A good example of pun is the well-known deer joke:

"-What do you call a deer with no eyes?
-No-eye deer",

in which the funniness comes from the homophony between the phrases "no eye deer" and "no idea".

One of the first and probably most robust systems in the field of pun processing is Binsted's JAPE – punning riddles generator (Binsted, 1996). Basing on a WordNet-related lexicon, it was able to generate quite a large spec of riddles – however, most of them were not evaluated highly by humans. Also worth mentioning is McKay's WISCRAIC system (McKay, 2002), generating simple puns in three different forms (question-answer, single sentence and two sentences sequence).

The main problem with most of existing humor generators is the simplicity and isolation of their output. Even if the jokes (riddles, puns) sound funny, it is still hard to imagine them in a natural surrounding, like daily dialogue. This in fact restricts possible applications of such systems – the best we can get is a system that tells jokes without any wider interaction context, just in isolated forms.

Actually, in the case of JAPE there was an attempt to integrate it into a system that interacts with users. Loehr implemented the system into a conversational agent Elmo, which interacted with players of an online role playing game (Loehr, 1996). The idea itself was quite interesting – however, the system's performance was evaluated lowly because the lack of relevance between users' utterances and the system's output.

1.3 Our contribution

In our research we decided to bridge this gap. We have developed a Japanese pun generating engine and implemented it into a chatterbot, creating a humor-equipped conversational system (Dybala et al., 2008). 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 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 that a non-humor-equipped one by the users. Also emotiveness analysis based evaluation (Dybala, 2009) showed that the system with humor actually made the users feel better and elicited more positive emotions than the one without humor.

In the research on joking chatterbots we used a simple pun generating engine, which based only on existing words (not phrases), that can be found in dictionaries. For example, for the word 父さん ("*tousan*" - father), it would generate a pun candidate 倒産 (*"tousan"* - bankruptcy), which is an existing word (one entrance in a dictionary), but it would not generate such candidates as 通さん (*"toosan"* – negative form of *toosu* – to let through), which is an inflected grammatical form. Similarly, with the previous system it is impossible to generate pun candidates which consist of more than one element.

In this paper we introduce a new, more complex algorithm that is able to generate not only existing words, but also phrasal candidates (see **2** for the outline).

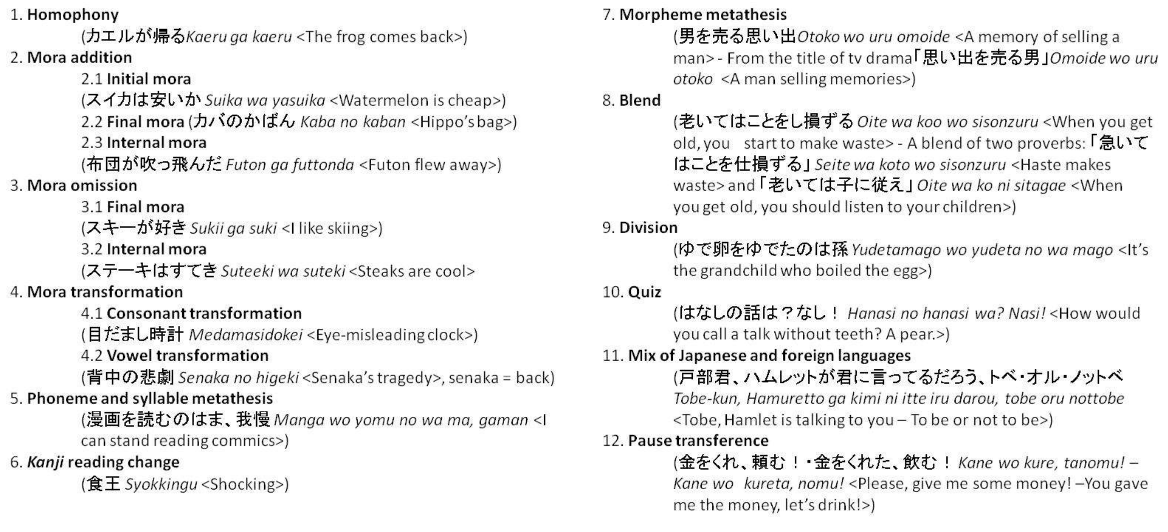
The evaluation experiment (see **3**) showed that the new algorithm generates pun candidates with 72.5% accuracy, which is promising enough to proceed with our research. Its next steps are outlined in **5**.

2 Pun generator

The outline of our pun generation algorithm is presented on Figure 2. In this section, we briefly describe its main parts.

2.1 Pun generation patterns

In our research, we base on a complex Japanese puns classification, proposed in our previous work (Dybala, 2006 – see Figure 1). A big corpus of human-created puns was built, and the puns were divided into 12 groups (with numerous subgroups), according to mora changes between the base phrases and phrases transformed into puns. For example, in a simple pun "*kaba no kaban*" ("hippo's bag"), the base phrase "*kaba*" (hippo) is transformed into *"kaban*" ("bag"). The technique used here is called "final mora addition", as there is one mora ("-*n*") added to the end of the base phrase.
The classification was used in our research to create pun generation patterns. For example, the group "final mora addition" 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 *karate* as an example).



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 1. Dybala's Japanese pun classification (with examples)

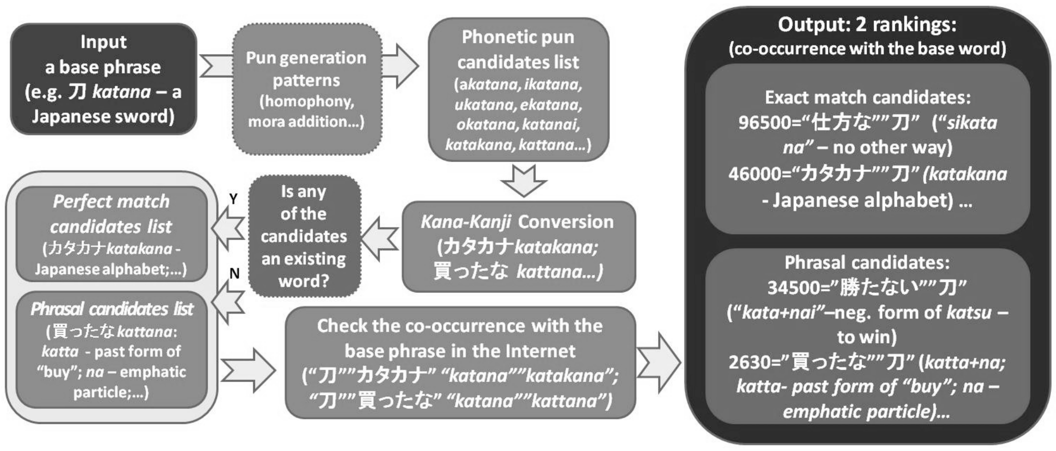

Input
a base phrase
(e.g. 刀 katana – a Japanese sword)
Pun generation patterns (homophony, mora addition...)
Phonetic pun candidates list (akatana, ikatana, ukatana, ekatana, okatana, katanai, katakana, kattana...)
Kana-Kanji Conversion (カタカナkatakana; 買ったな kattana...)
Is any of the candidates an existing word?
Y
N
Perfect match candidates list (カタカナkatakana - Japanese alphabet;...)
Phrasal candidates list (買ったなkattana: katta - past form of "buy"; na – emphatic particle;...)
Check the co-occurrence with the base phrase in the Internet ("刀""カタカナ" "katana""katakana"; "刀""買ったな" "katana""kattana")
Output: 2 rankings:
(co-occurrence with the base word)
Exact match candidates:
96500="仕方な""刀" ("sikata na" – no other way)
46000="カタカナ""刀" (katakana - Japanese alphabet) ...
Phrasal candidates:
34500="勝たない""刀" ("kata+nai"–neg. form of katsu – to win)
2630="買ったな""刀" (katta+na; katta- past form of "buy"; na – emphatic particle)...

Figure 2. Pun candidates generating algorithm flowchart

1. **homophony** ([*karate*])
2. **initial mora addition** (**karate*: *akarate, ikarate, ukarate...*)
3. **internal mora addition** (*ka*rate, kara*te: karaate, karaite, karaute...*)
4. **final mora addition** (*karate*: karatea, karatei, karateu...*)
5. **final mora omission** (*kara*)
6. **internal mora omission** (*kate*)
7. **mora transformation** (*garate, tarate, harate...*)

In the 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. (1996) to find phrases that sound more similar than others.

The patterns were used in the algorithm to generate phonetic candidates list – see **2.2**.

2.2 Pun candidates generation algorithm

Figure 2 presents the outline of our pun generating system.

Its 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 (usually a noun, a verb or an adjective).

In the next step, the base phrase is transformed using the pun generation patterns (see **2.1**), in order to generate a list of phonetic candidates. To this point, everything happens in *Hiragana* (Japanese syllable alphabet). Next, each of the candidates is converted into *Kanji* (Chinese characters), using MeCab-skkserv *Kana-Kanji* Converter[1]. In most of the cases, though, there is more than one transcription possible – thus, we decided to let the system generate two different options everywhere when it was possible. There are cases in which even more conversions is possible – however, any further increase of possible transcriptions would give us an enormous amount of queries to be made, which, in turn, is quite time-consuming.

In the next step, all of converted candidates are analyzed by POS and morphological analyzer MeCab[2] 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 phrasal candidates.

Next, candidates from both lists are used to perform multiple queries in the Internet (currently - Yahoo search engine). The reason we decided to use the Internet instead of hand-made lexicons or existing sources (as in other pun generators, i.e. JAPE) is that the dictionaries that already exist are often not adaptable to the purpose of joke generation, and building our own, pun-generation oriented lexicon would be quite laborious. The Internet, though, constantly being updated, it contains the most current information, also about the language, and can be easily accessed – all we need is a good algorithm that would allow us to use these vast resources.

Thus, in our research we decided to use the Internet to extract lexical information about the pun candidates. The pun's base phrase must be somewhat related to the phrase that will be used in a pun (i.e., they cannot be completely irrelevant; otherwise all we will get will be abstract humor). Therefore, our algorithm checks the co-occurrence of each candidate with the base phrase. If, for example, the base phrase is *katana*, and one of the candidates is *kattana*, the query phrase for Yahoo is <"*katana*" "*kattana*">.

[1] MeCab-skkserv *Kanji-Kana* converter, http://chasen.org/~taku/software/mecab-skkserv/

[2] MeCab: Yet another part-of-speech and morphological analyzer, T.Kudo, http://mecab.sourceforge.ne

Human-created joke: 白菜は臭い. **(*Hakusai ha kusai* – Chinese cabbage stinks)**
base phrase: 白菜 (***hakusai*** - Chinese cabbage) – system input

results – co-occurrence with the base phrase ranking:

-exact match candidates (<co-occurrence hit rate>=<proposition>):
<19600>=<北斎> (*hokusai* – Japanese name; famous Japanese painter)
<70>=<舶載> (*hakusai* - ocean transportation; importation)

-phrasal candidates (<co-occurrence hit rate>=<proposition>):
<34700>=<は+臭い> (*ha+kusai; ha* – subject particle; *kusai* – to stink)
<750>=<歯+臭い> (*ha+kusai; ha* – tooth/teeth; *kusai* – to stink)
<400>=<は+救済> (*ha+kyuusai; ha* – subject particle; *kyuusai* – help, salvation)
<91>=<は+ψ> (*ha+pusai; ha* – subject particle; *pusai* – letter ψ)
<26>=<白+タイ> (*haku+tai; haku* – white; *tai* – Thailand)
<13>=<履く+際> (*haku+sai; haku* – put on, wear; *sai* – time, moment of time; when)
<11>=<吐く+際> (*haku+sai; haku* – to vomit; *sai* – moment of time; when)
<5>=<白+さえ> (*haku+sae; haku* – white; *sae* – if only)
<2>=<画伯+祭> (*gahaku+sai; gahaku* - master painter; *sai* – moment of time; when)
<1>=<母+臭い> (*haha+kusai; haha* – mother; *kusai* – to stink)

Figure 3. System's performance – results for the input *hakusai* (Chinese cabbage)

Next, for both lists (perfect match candidates and phrasal candidates) a ranking is formed, starting from the candidate that had the highest co- occurrence rate with the base phrase.

Figure 3 presents the results for the base phrase "*hakusai*" (a Chinese cabbage), taken from the human-created joke "*Hakusai ha kusai"* (Chinese cabbage stinks).

3 Evaluation Experiment

To check if our approach is right, we conducted an evaluation experiment.

In our previous experiments with joking chatterbot we used an algorithm that generated only words (not phrasal) candidates. Thus, this time we also wanted to check how the implementation of phrasal pun candidates generation algorithm changed the system's overall performance. Therefore, we compared the results for the system before and after adding the phrase generation option.

Evaluation of such entities as humor is always quite troublesome and there is no robust methodology in this field. In our experiment, we tried to compare the system's performance to the level of human. The details are described below.

Experiment aims:
To test the system's performance and compare it with the one we used in our previous research.

Experiment method:
Compare the system's performance in pun generating to the level of human. Calculate the results and compare them with those from our previous experiments.

Experiment materials:
200 human-created jokes, chosen from a Japanese puns database (Sjöbergh et al., 2008).

Experiment procedure:
From the 200 human-created jokes, we have extracted base phrases, which were then used as input for our system. Next, we checked if among the candidates generated for each phrase was the word actually used in the human-created pun. We calculated the percentage of situations in which there was a match between the system's results and human-created puns, and compared the result with that from our previous experiment (without phrase generation).

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 a right candidate.

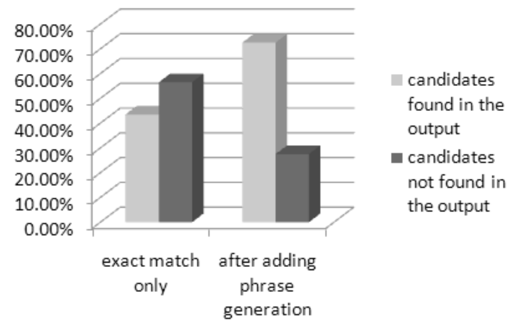


Figure 4. Evaluation experiment results

Experiment results:
As showed on Figure 4, when only exact match candidates generation was used, the system was successful only in 43.5% cases, which is not a very impressing result. However, after adding the

phrasal candidates generation part, 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.

4 Discussion

The evaluation experiment was fairly successful and showed that in most cases the algorithm can generate proper pun candidates, also these based on phrases, not only on single words.

4.1 What were the rankings for?

Actually, before constructing the algorithm, we made one more assumption. As there must be some kind of correlation between the base phrase and the word used in the pun (as *katana* and *katta na*), we assumed that checking the co-occurrence of these two and making a ranking of candidates will allow us to see some regularities. If, for instance, the phrase that was actually used in the human-created joke was always on the top or near the top, this would allow us to choose the best candidate from the list or at least to reduce their number. This, however, did not turn out to be true – the positions of candidates that were used in human-created puns varied for every joke, from the top to the very bottom of the ranking. Therefore, the co-occurrence ranking alone is not enough to choose the right candidate.

4.2 Human choice = the best choice?

As mentioned above, in our evaluation experiment 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 generated 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 3) 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 歯臭い (*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.

Therefore, we believe that – with a proper set of sentence integration rules – most of candidates can be used to form a pun. An idea for checking if this approach is right is presented in **5.4**.

5 Conclusion and future work

In this paper we presented a system that generates pun candidates for given base phrases. It can generate not only single words, but also more complex phrases. The system's performance was tested in the evaluation experiment. Its results are promising enough to justify further research on this subject. Furthermore, we proved that the Internet is a good and usable source for pun generation – this will surely spare the researchers some time, as there is no need to manually create large humor-oriented dictionaries.

Implementation of the system into a chatterbot should visibly improve the results of our previous research (Dybala et al., 2008, 2009). To achieve that, some work is still to be done – in particular, in the nearest future we are planning to focus on two major tasks: filtrating the candidates list and sentence integration algorithm.

5.1 Filtering the list of candidates

One of the potential problems noticed during the experiment was the length of the candidates list, which, in some cases, exceeded 100 propositions. Although we are very happy to see the system working that prolifically, and we assume that in most cases there is no one and only one right answer, on some (especially very long) lists some candidates are obviously better that others. Therefore, we need a good algorithm to filter the candidate list and reduce it when too long. One possible idea to do that is to create a hierarchy based on the type of phonetic generation pattern used (see **2.1**). For example, homophony or mora addition lead to generation of phrases that are more similar to the base phrase than those generated using mora omission. This idea is currently being tested, with some very promising results of preliminary experiments.

5.2 Sentence integration algorithm

In our research on joking chatterbots, we used pun generation algorithm that generated only single words (not phrasal) candidates, which were easier to integrate into sentence forms, as they do not include any inflected forms and in most cases are nouns, uninflected verbs or adjectives. To generate pun-including sentences, in the previous system we used such patterns as:

[base phrase] *to ieba* [candidate] *desu ne*

(speaking of [base phrase], it's [candidate]),

which worked quite well for the single words. However, in the case of phrases, such candidates as *ha kusai* (topic particle + stinks) are much more difficult to insert into such patterns due to their grammatical complexity. Thus, here we face a serious problem: either we create a large set of sentence integration patterns, to cover as many possibilities as possible (which would be quite laborious), or again use the power of Internet, this time to find what sentences that actually include the candidate phrase look like, and extract patterns that would fit that particular candidate. For example, for the candidate 歯臭い (*ha kusai* – teeth stink), the system could find a sentence pattern [noun]を食べると歯臭い ([noun]*wo taberu to ha kusai* – if you eat [noun], your teeth stink). Next, since the base phrase is a noun, the pattern can be used to form a punning sentence: 白菜を食べると歯臭い (*hakusai wo taberu to ha kusai* – if you eat Chinese cabbage, your teeth stink). Such approach would spare us the effort of creating the whole set of patterns manually, and allow the system to always generate appropriate pattern for the particular pair of candidate and base phrase.

5.3 Time issues

As we use the Internet search engines (currently – Yahoo), the amount and frequency of allowed queries is restricted and if the system puts too much load on the search engine, it is recognized as a spammer and temporarily blocked. Therefore, we had to put pauses between each query, which caused quite a severe extension of time needed to generate the candidates. The time differs according to 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. Needless to say, this problem has to be solved in the nearest future – we are planning to construct a large Internet-based text corpus, which will allow us to perform the queries offline.

5.4 Evaluation of jokes

Evaluation of such phenomena as humor is always a troublesome issue. In this paper we described a method that allowed us to check if the system is able to generate pun candidates in a similar way that humans do. This approach, however, is good only on the level of candidates generation. Evaluating the usability of generated candidates is much more difficult and by definition subjective, as it requires the presence of human evaluators. If, for example, in order to evaluate the candidates usability, you show the evaluators the list and ask them to tell which one is better, their assessment would be obviously limited by their imagination. 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.

One possible way to solve this problem on the level of candidate usefulness evaluation, we could use a sort of Wizzard-of-Oz approach: all candidates could be manually integrated into punning sentences (using as simple patterns as possible), and then showed to the evaluators as complete puns. This method is still to be tested – however, we believe it could be successful to assess the usefulness of candidates and, in consequence, lead to construction of more sophisticated pun generation algorithm.

References

Kim Binsted. 1996. *Machine humour: An implemented model of puns*. Univ. of Edinburgh, UK.

Kim Binsted, and Osamu Takizawa. 1997. *Computer generation of puns in Japanese.* Sony Computer Science Lab, Communications Research Lab.

Arnie Cann, Kitty Holt, and Lawrence G. Calhoun. 1999. The roles of humor and sense of humor in responses to stressors. *Humor: International Journal of Humor Research*, 12(2):. 177–193.

Karen S. Cook, and Eric Rice. 2003. Social exchange theory. J. Delamater (Ed.), *Handbook of social psychology*, 53--76. NewYork: Plenum (2003) 11. Difficult subjects

Pawel Dybala. 2006.*Dajare - Nihongo ni okeru dōon'igi ni motozuku gengo yūgi (Dajare – Japanese puns based on homophony)*. Jagiellonian Univ., Press, Kraków, Poland.

Pawel Dybala, Michal Ptaszynski, Shunsuke Higuchi, Rafal Rzepka, and Kenji Araki. 2008. *Humor Prevails! - Implementing a Joke Generator into a Conversational System*. In the Proceedings of the 21st Australasian Joint Conference on AI (AI-08), Wobcke, W. and Zhang, M. (eds), Auckland, New Zealand, 2008. Springer-Verlag LNAI Vol. 5360, 214-225, Springer Berlin & Heidelberg.

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka, and Kenji Araki 2009. *Humorized Computational Intelligence - towards User-Adapted Systems with a Sense of Humor*. In the Proceedings of the EvoStar 2009 Conference, EvoWorkshops. M. Giacobini et al. (Eds.). Springer-Verlag LNCS, Vol. 5484, 452–461, Springer Berlin & Heidelberg

Dan Loehr. 1996. *An integration of a pun generator with a natural language robot*, Proc. Intern. Workshop on Computational Humor, University of Twente, Netherlands, 161-172.

Justin McKay. 2002. *Generation of idiom-based witticisms to aid second language learning*. In: Stock et al., 77--87

Carmen C. Moran. 1996. Short-term mood change, perceived funniness, and the effect of humor stimuli.' *Behavioral Medicine 22*(1): 32—38

Michael Mulkay. 1988. *On humor: Its nature and its place in modern society*. NY: Basil Blackwell

Susan Sprecher, and Pamela C. Regan. 2002. Liking some things (in some people) more than others: Partner preferences in romanic relationships and friendships. *Journal of Social and Personal Relationships*, 19(4): 463—481

Jonas Sjöbergh, and Kenji Araki. 2008. *Robots Make Things Funnier*, Proceedings of LIBM'08, Asahikawa, Japan

Osamu Takizawa, Masuzo Yanagida, Akira Ito, and Hitoshi Isahara. 1996. *On computational processing of rhetorical expressions | puns, ironies and tautologies*. In Proc. of The International Workshop on Computational Humor, 39-52, Netherland