Data Filtering in Humor Generation: Comparative Analysis of Hit Rate and Co-occurrence Rankings as a Method to Choose Usable Pun Candidates

Pawel Dybala
JSPS Research Fellow / Otaru University of Commerce, Midori 3-5-21, 047-8501 Otaru, Japan.
paweldybala@res.otaru-uc.ac.jp

Rafal Rzepka
Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, 060-0814 Sapporo, Japan.
kabura@media.eng.hokudai.ac.jp

Kenji Araki
Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, 060-0814 Sapporo, Japan.
araki@media.eng.hokudai.ac.jp

Kohichi Sayama
Otaru University of Commerce, Department of Information and Management Science, Midori 3-5-21, 047-8501 Otaru, Japan.
sayama@res.otaru-uc.ac.jp

ABSTRACT

In this paper we propose a method of filtering excessive amount of textual data acquired from the Internet. In our research on pun generation in Japanese we experienced problems with extensively long data processing time, caused by the amount of phonetic candidates generated (i.e. phrases that can be used to generate actual puns) by our system. Simple, naive approach in which we take into considerations only phrases with the highest occurrence in the Internet, can effect in deletion of those candidates that are actually usable. Thus, we propose a data filtering method in which we compare two Internet-based rankings: a co-occurrence ranking and a hit rate ranking, and select only candidates which occupy the same or similar positions in these rankings. In this work we analyze the effects of such data reduction, considering 1 cases: when the candidates are on exactly the same positions in both rankings, and when their positions differ by 1, 2, 3 and 4. The analysis is conducted on data acquired by comparing pun candidates generated by the system (and filtered with our method) with phrases that were actually used in puns created by humans. The results show that the proposed method can be used to filter excessive amounts of textual data acquired from the Internet.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval - *Information filtering*

General Terms: Algorithms, Performance, Experimentation

Keywords: humor processing, web-based data extraction, AI, NLP, HCI.

1. INTRODUCTION

Humor processing is a field of science that joins such areas as AI (Artificial Intelligence), NLP (Natural Language Processing) or HCI (Humor-Computer Interaction). In our research project (summarized in [1]) we aim to construct a conversational system able to generate humor during interactions with users. As humor itself is a very complex phenomenon, in our research we decided to narrow the focus to its linguistic genre, known as "puns". The research is conducted in Japanese (however, we believe that most of algorithms and resources we develop are language independent and can be relatively easily transferred to other languages).

Japanese puns, called *dajare*, are mostly based on word homophony. Like puns in most other languages, they are often told in conversations unexpectedly, i.e. without any announcement. This sometimes requires a long introduction, told in order to misguide the partner and provide a proper pun context. However, as context processing and generation is still a problematic field in NLP, in our research we focus on simple one-liner puns. One simple example of such pun is *Kaeru ga kaeru* (The frog comes back), based on two different meanings of the word *kaeru* ("a frog" and "come back"). Such puns can be incorporated into conversation by using a word from the interlocutor's previous utterance. The above example can be an answer to such utterance as "I saw a frog today". This would require using some punning-response templates, as those proposed in our earlier works [1] (e.g. "Speaking of frogs, ...").

In our research we use web search engines to generate pun candidates, i.e. phrases that can be used to generate puns. The phonetic candidates generation is based on a complex Japanese puns techniques classification, which leads to generating great amounts of candidates. Each of them is queried in the Internet to check its hit rate and co-occurrence with the base phrase (phrase that is converted into a pun). Basing on the results of these two types of queries, two rankings of candidates are formed: a hit-rate ranking and a co-occurrence ranking. The lists of candidates acquired in this process, however, are quite long and include many phrases that do not seem usable for pun generation. Thus, we decided to filter the candidates to select those possibly usable and delete those unnecessary. To do that, we proposed a filtering method in which we compare the two above mentioned rankings.

In below sections we first briefly describe the algorithm of our pun generator (2), and explain the problem with extensive data generation and insufficiency of naive approaches in filtering. Next, we describe the method proposed to solve this problem (3). The method was tested experimentally (4). The results are concluded (5) and some directions for the future are given.

2. PUN GENERATOR

The aim of our research project is to construct a pun generator for Japanese and implement it into a chatterbot. Since it was already showed that humor can facilitate HCI (see i.e. [2] or [3]) we assumed that building a system able to generate linguistic jokes (puns) during conversations with users is a worthwhile task.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

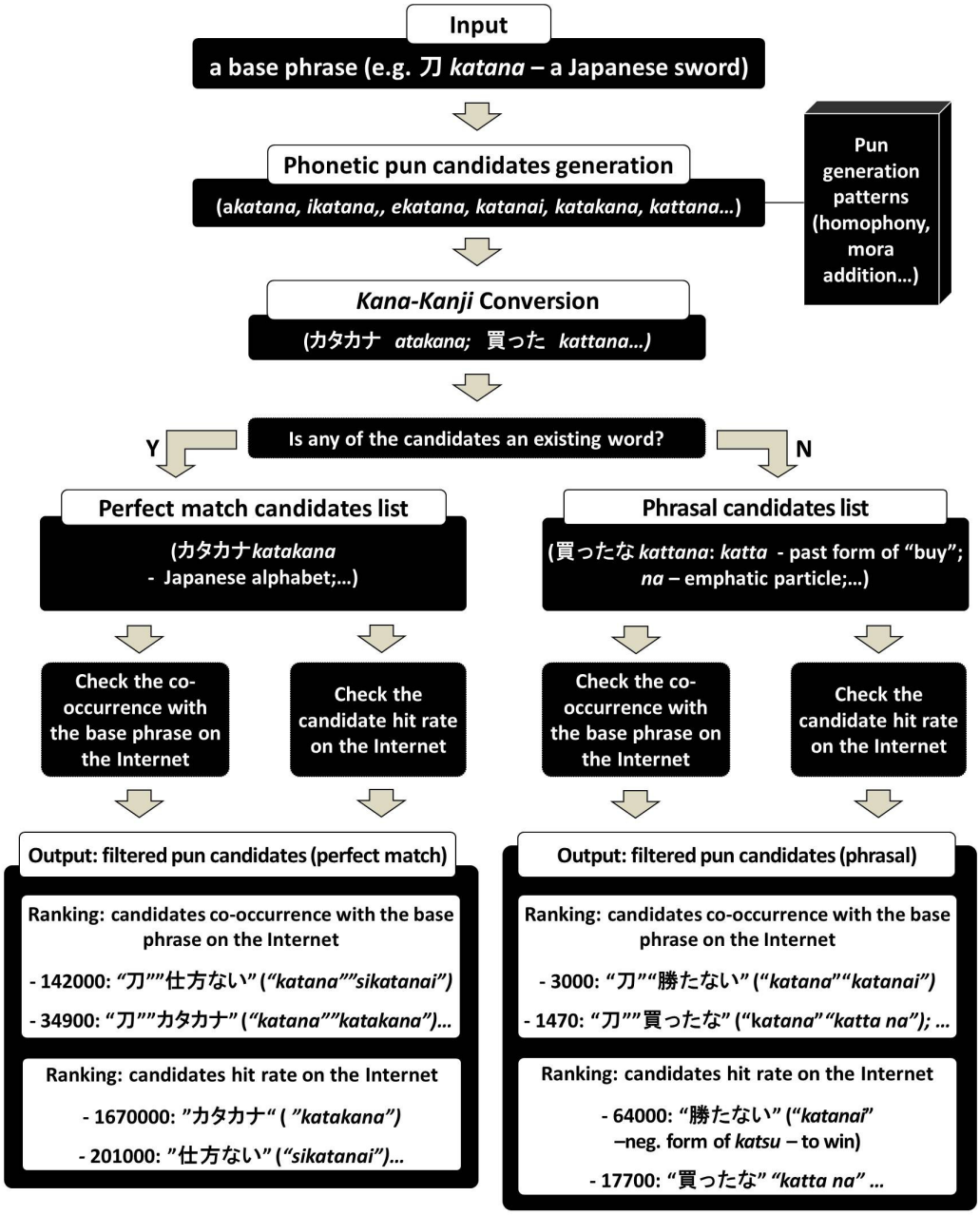

Figure 1. Pun candidates generation algorithm.

In our earlier works, we presented a very simple, single-word-pun generator implemented into a chatterbot. The evaluation results showed that even not very high leveled puns can enhance conversations between users and system. In numerous experiments, users evaluated the pun telling chatterbot as better, more interesting and making them feel more positive than a similar system without sense of humor [4].

As mentioned above, the system developed in our previous research project did generate puns, albeit not very sophisticated nor funny. Thus, in the next stage of our project we decided to develop a more complex pun generator, which generates not only simple word, but also phrase puns, i.e. puns consisting of more than one single word (see [5] for details).

The pun generation in our systems is based on phonetic classification of Japanese puns, proposed in our earlier work [6]. In this classification, 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 [6].

The classification was used in our research to create pun generation patterns. For example, the above mentioned group "final mora addition" gives us the pattern "[base phrase]+[*]", where [*] means one single mora. Currently, there are seven

patterns implemented in the system: homophony, initial mora addition, internal mora addition, final mora addition, final mora omission, internal mora omission, and mora transformation. In the last pattern the number of possible transformations is very large, therefore, in our system we used Japanese phoneme similarity values, proposed by Takizawa et al. [7] to identify phrases that sound more similar than others.

The pun generation patterns were used in the pun candidates generation algorithm. Its outline is showed on Figure 1 on an example of a word *katana* (a Japanese sword). From an input, which is a sentence or a phrase, the system extracts base phrase, i.e. a phrase that will be transformed into a pun, usually a noun or adjective. Next, after converting the base phrase into Japanese *Kanji* characters, the system checks if any of converted phrases is an existing word. To do that, it uses the MeCab POS Analyser [8]. All existing words form the perfect match candidates list, and all remaining words form the phrasal candidates list. Next, phrases from both of these lists are queried in the Internet. The initial concept was to use the Web to check each pun candidate's co-occurrence with the base phrase and simply select the 5 or 10 top ones to be used in the next processing stages, in which the system would chose one best candidate to generate a pun. Initial experiments, however, showed that such an approach gives rather unsatisfying results. Thus, we tested another method, in which we selected only the 5 or 10 candidates with the highest hit rate in the Internet (no co-occurrence). This approach was already used in one of our earlier works and worked reasonably well in generating simple puns [1]. However, in phrasal puns generation simple Internet hit rate was also not enough to select usable candidates and filter out those unnecessary. On the other hand, using the whole generated lists in the next processing stages (such as sentence level context integration), which will also be based on the Internet, would significantly slow the processing time, as in some cases the lists may even include over 150 candidates. The final goal of this research is implementation of the pun generator into chatterbot, performing real time conversations with users, and extensive processing time might seriously hinder this process. Thus, we directed our efforts on developing an algorithm which would allow to select usable candidates from the list and reduce their number at the same time, with possibly minimal decrease of the system's accuracy.

3. PROPOSED METHOD

The above mentioned findings that simple choose of candidates with the highest hit rates or co-occurrence with the base phrase are consistent with claims of humor specialists, such as Suls [9] or Ruch [10]. They state that humor should rather base on surprise and mismatch between expectations and actual joke than on joining concepts that are commonly associated with each other. Thus, choosing only the phrases that are either the most popular in the Internet or occur most often with the base phrase may not always be efficient according to this definition.

That said, it does not mean that the words that form a pun should be not related to each other and not popular at all, as such an approach would rather lead to generating a very abstract and even senseless humor. Therefore we decided to conduct experiments using the two above mention methods: co-occurrence with the base phrase check and hit rate check. We compared positions of each candidate in these two rankings to investigate what are the chances to select usable pun candidates by choosing only those which occupy the same or similar positions. To do so, we used base phrases from human created puns.

4. EXPERIMENT

From a Japanese human-created pun database [11] we chose 200 jokes and used their base phrases as our system's input, making it generate as many candidates as possible, and form them into the two rankings. Then, we compared the positions of phrases that were actually used in the jokes in both of these rankings. For example, from the joke: *Katana wo katta na* ("Bought a Japanese sword, you know") we extracted the base phrase *katana* and used it as an input for our system. Then, we compared the positions of "*kattana*" in the hit rate ranking and *"katana"+ "kattana"* in the co-occurrence ranking.

Such an analysis was conducted for all 200 puns. By doing so, we investigated the probabilities of selecting usable pun candidates with the proposed method. We took into consideration not only candidates that occupy exactly the same positions in both rankings, but also those whose positions differ by 1, 2, 3 and 4. As we aimed at reducing the number of candidates, we also analyzed the percentage of its reduction in this process. The results are presented in Table 1.

Table 1. Correlation between the probability of finding the candidate (P_f) from human-created pun on the list after reduction and the average reduction rate of candidates (R_r)

	exact	**+-1**	**+-2**	**+-3**	**+-4**
P_f	46.2%	72.4%	80.7%	87.6%	89.5%
R_r	70.6%	46.8%	33.3%	27.9%	21.6%

Figure 2 shows the results of ranking comparison for the base phrase *kaimono* (shopping), taken from the joke *"kaimono wa wakai mono ni tanomou"* ("Let's ask the young people to do the shopping"), for phrasal candidates.

The results of this analysis showed that the probability of finding the candidates (P_f) used in human created puns with the proposed method is 46.2% when we take into consideration only the candidates that occupy the same positions in both rankings. In this case, the average reduction of the number of candidates (R_r) is 70.5%. If we take into consideration also candidates which positions differ by 1, P_f is 72.4%, and R_r is 46.8%. If the candidates' positions differ by 2, P_f rises to 80.7% and R_r drops to 33.3%. If the candidates differ by 3, P_f is 87.6% and R_r is 27.9%. Taking into consideration candidates whose positions differ by 4 rises P_f to 89.5%, while R_r drops to 21.6%.

5. CONCLUSIONS

As showed above, the proposed method can be used to filter pun candidates lists with a fairly successful percentage when we take into consideration candidates whose positions differ by 1 or more. Since one of the aims of developing this method was to reduce the number of candidates, it seems reasonable to take into consideration candidates that differ by 1 or 2 positions and use them in further processing stages. However, as the number of generated candidates differs in each case, instead of setting a rigid rule (like "max. positions difference = 2"), we are planning to let the system decide to what extent it should reduce the number of candidates (i.e. "if candidates number < X, max. position difference = 1; if candidates number > X, max. position difference = 2" etc., depending also on the equipment the system will be working on at the time).

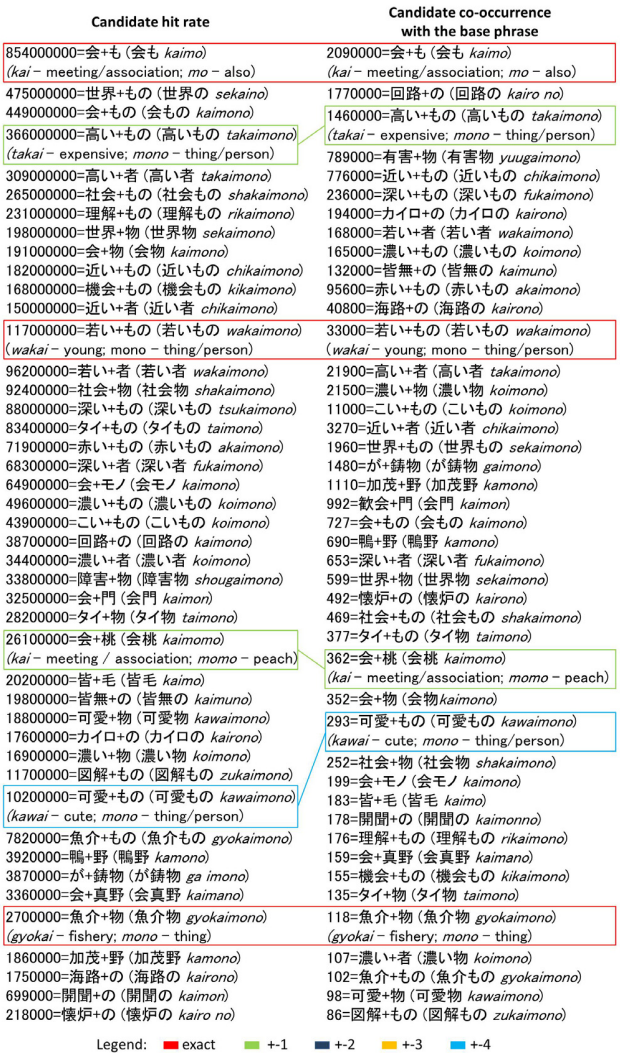

Figure 2. Output example for the word *kaimono* (shopping) – phrasal match candidates lists with marked positions

The results described in this paper should be useful not only in research on humor processing. The rankings comparison method, proposed above, seems to be an effective tool in textual data filtering, when there is a need to select phrases related to each other in particular manner. In the nearest future we are planning to test this method in metaphor generation in Japanese.

As mentioned above, in the next stage of the algorithm, the system will select one candidate that will be used to generate a pun. To do that, we are planning to use the approach proposed by Shen and Engelmayer, who experimentally showed that the salience imbalance theory, commonly used in metaphor processing, can be also used to process humor (see [12] for details). Currently we are working on optimal methods of calculating such salience degrees that would allow the system to select the best pun candidates.

The proposed method can be used to generate puns in the real time during conversations with users. However, it might be more efficient to store the generated candidates to use them in an offline manner to avoid time lags during conversations with users. As the Internet changes constantly, such data would need to be updated frequently.

The data filtering method proposed in this paper was developed for Japanese. However, as it is mostly Internet based and uses an offline dictionary in a very limited extent, we believe that it should be applicable also to other languages.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

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

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

[3] Morkes, J., Kernal, H. K., Nass, C. 1999. Effects of humor in task-oriented human-computer interaction and computer-mediated communication: A direct test of srct theory. *Human-Computer Interaction*, vol. 14, no. 4, 395-435.

[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. *Journal of Ambient Intelligence and Smart Environments 2,* 31-48.

[5] Dybala, P., Ptaszynski, M., Rzepka, R., Araki, K. 2009. Crossing Word Borders: Towards Phrasal Pun Generation Engine. In Proceedings of PACLING 2009 (Sapporo, Japan, 2009), 242-247.

[6] Dybala, P. 2006. *Dajare - Nihongo ni okeru do'on igi ni motozuku gengo yugi* (Japanese puns based on homophony). Krakow, Poland: Jagiellonian University, MA Dissertation.

[7] Takizawa, O., Yanagida, M., Ito, A., Isahara, H. 1996. On computational processing of rhetorical expressions: puns, ironies and tautologies. In: *Proc. of The International Workshop on Computational Humor* (Netherlands, 1996), 39-52.

[8] Kudo, T. 2001. MeCab: Yet another part-of-speech and morphological analyzer, http://mecab.sourceforge.ne

[9] Suls, J. 1972. A two-stage model for the appreciation of jokes and cartoons: an information processing analysis. In *The Psychology of Humor*, J.H. and McGhee P.E. (eds.), New-York: Academic Press, pp. 81-100.

[10] Ruch, W. 2001. The perception of humor. In *Emotion, Qualia, and Consciousness*, A. Kaszniak (ed.), Word scientific publisher, Tokyo, pp 410-425.

[11] Sjöbergh, J., and Araki, K. 2008. Robots Make Things Funnier. In: *Proceedings of LIBM'08* (Asahikawa, Japan, 2008), pp. 46-51.

[12] Shen, Y. and 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 (2012) (available at: http://www.tau.ac.il/~yshen/publications/%20friend%20is%20like%20an%20anchor.pdf