Unsupervised joke generation from big data

Saša Petrović
School of Informatics
University of Edinburgh
sasa.petrovic@ed.ac.uk

David Matthews
School of Informatics
University of Edinburgh
dave.matthews@ed.ac.uk

Abstract

Humor generation is a very hard problem. It is difficult to say exactly what makes a joke funny, and solving this problem algorithmically is assumed to require deep semantic understanding, as well as cultural and other contextual cues. We depart from previous work that tries to model this knowledge using ad-hoc manually created databases and labeled training examples. Instead we present a model that uses large amounts of unannotated data to generate *I like my X like I like my Y, Z* jokes, where X, Y, and Z are variables to be filled in. This is, to the best of our knowledge, the first fully unsupervised humor generation system. Our model significantly outperforms a competitive baseline and generates funny jokes 16% of the time, compared to 33% for human-generated jokes.

1 Introduction

Generating jokes is typically considered to be a very hard natural language problem, as it implies a deep semantic and often cultural understanding of text. We deal with generating a particular type of joke – *I like my X like I like my Y, Z* – where X and Y are nouns and Z is typically an attribute that describes X and Y. An example of such a joke is *I like my men like I like my tea, hot and British* – these jokes are very popular online.

While this particular type of joke is not interesting from a purely generational point of view (the syntactic structure is fixed), the content selection problem is very challenging. Indeed, most of the X, Y, and Z triples, when used in the context of this joke, will not be considered funny. Thus, the main challenge in this work is to "fill in" the slots in the joke template in a way that the whole phrase is considered funny.

Unlike the previous work in humor generation, we do not rely on labeled training data or hand-coded rules, but instead on large quantities of unannotated data. We present a machine learning model that expresses our assumptions about what makes these types of jokes funny and show that by using this fairly simple model and large quantities of data, we are able to generate jokes that are considered funny by human raters in 16% of cases.

The main contribution of this paper is, to the best of our knowledge, the first fully unsupervised joke generation system. We rely only on large quantities of unlabeled data, suggesting that generating jokes does not always require deep semantic understanding, as usually thought.

2 Related Work

Related work on computational humor can be divided into two classes: humor recognition and humor generation. Humor recognition includes double entendre identification in the form of *That's what she said* jokes (Kiddon and Brun, 2011), sarcastic sentence identification (Davidov et al., 2010), and one-liner joke recognition (Mihalcea and Strapparava, 2005). All this previous work uses labeled training data. Kiddon and Brun (2011) use a supervised classifier (SVM) trained on 4,000 labeled examples, while Davidov et al. (2010) and Mihalcea and Strapparava (2005) both use a small amount of training data followed by a bootstrapping step to gather more.

Examples of work on humor generation include dirty joke telling robots (Sjöbergh and Araki, 2008), a generative model of two-liner jokes (Labutov and Lipson, 2012), and a model of punning riddles (Binsted and Ritchie, 1994). Again, all this work uses supervision in some form: Sjöbergh and Araki (2008) use only human jokes collected from various sources, Labutov and Lipson (2012) use a supervised approach to learn feasible circuits that connect two concepts in a semantic network, and

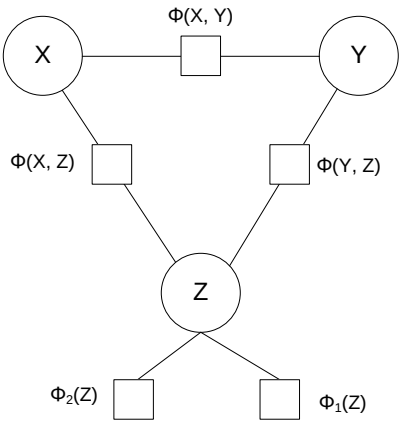

Figure 1: Our model presented as a factor graph.

Binsted and Ritchie (1994) have a set of six hard-coded rules for generating puns.

3 Generating jokes

We generate jokes of the form *I like my X like I like my Y, Z*, and we assume that X and Y are nouns, and that Z is an adjective.

3.1 Model

Our model encodes four main assumptions about *I like my* jokes: i) a joke is funnier the more often the attribute is used to describe both nouns, ii) a joke is funnier the less common the attribute is, iii) a joke is funnier the more ambiguous the attribute is, and iv) a joke is funnier the more dissimilar the two nouns are. A graphical representation of our model in the form of a factor graph is shown in Figure 1. Variables, denoted by circles, and factors, denoted by squares, define potential functions involving the variables they are connected to.

Assumption i) is the most straightforward, and is expressed through $\phi(X, Z)$ and $\phi(Y, Z)$ factors. Mathematically, this assumption is expressed as:

$$\phi(x, z) = p(x, z) = \frac{f(x, z)}{\sum_{x,z} f(x, z)}, \qquad (1)$$

where $f(x, z)$[1] is a function that measures the co-occurrence between x and z. In this work we simply use frequency of co-occurrence of x and z in some large corpus, but other functions, e.g., TF-IDF weighted frequency, could also be used. The same formula is used for $\phi(Y, Z)$, only with different variables. Because this factor measures the similarity between nouns and attributes, we will also sometimes refer to it as *noun-attribute similarity*.

Assumption ii) says that jokes are funnier if the attribute used is less common. For example, there are a few attributes that are very common and can be used to describe almost anything (e.g., new, free, good), but using them would probably lead to bad jokes. We posit that the less common the attribute Z is, the more likely it is to lead to surprisal, which is known to contribute to the funniness of jokes. We express this assumption in the factor $\phi_1(Z)$:

$$\phi_1(z) = 1/f(z) \qquad (2)$$

where $f(z)$ is the number of times attribute z appears in some external corpus. We will refer to this factor as *attribute surprisal*.

Assumption iii) says that more ambiguous attributes lead to funnier jokes. This is based on the observation that the humor often stems from the fact that the attribute is used in one sense when describing noun x, and in a different sense when describing noun y. This assumption is expressed in $\phi_2(Z)$ as:

$$\phi_2(z) = 1/senses(z) \qquad (3)$$

where $senses(z)$ is the number of different senses that attribute z has. Note that this does not exactly capture the fact that z should be used in different senses for the different nouns, but it is a reasonable first approximation. We refer to this factor as *attribute ambiguity*.

Finally, assumption iv) says that dissimilar nouns lead to funnier jokes. For example, if the two nouns are *girls* and *boys*, we could easily find many attributes that both nouns share. However, since the two nouns are very similar, the effect of surprisal would diminish as the observer would expect us to find an attribute that can describe both nouns well. We therefore use $\phi(X, Y)$ to encourage dissimilarity between the two nouns:

$$\phi(x, y) = 1/sim(x, y), \qquad (4)$$

where sim is a similarity function that measures how similar nouns x and y are. We call this factor *noun dissimilarity*. There are many similarity functions proposed in the literature, see e.g., Weeds et al. (2004); we use the cosine between the distributional representation of the nouns:

$$sim(x, y) = \frac{\sum_z p(z|x)p(z|y)}{\sqrt{\sum_z p(z|x)^2 * \sum_z p(z|y)^2}} \qquad (5)$$

[1] We use uppercase to denote random variables, and lowercase to denote random variables taking on a specific value.

Equation 5 computes the similarity between the nouns by representing them in the space of all attributes used to describe them, and then taking the cosine of the angle between the noun vectors in this representation.

To obtain the joint probability for an (x, y, z) triple we simply multiply all the factors and normalize over all the triples.

4 Data

For estimating $f(x, y)$ and $f(z)$, we use Google n-gram data (Michel et al., 2010), in particular the Google 2-grams. We tag each word in the 2-grams with the part-of-speech (POS) tag that corresponds to the most common POS tag associated with that word in Wordnet (Fellbaum, 1998). Once we have the POS-tagged Google 2-gram data, we extract all (noun, adjective) pairs and use their counts to estimate both $f(x, z)$ and $f(y, z)$. We discard 2-grams whose count in the Google data is less than 1000. After filtering we are left with 2 million (noun, adjective) pairs. We estimate $f(z)$ by summing the counts of all Google 2-grams that contain that particular z. We obtain $senses(z)$ from Wordnet, which contains the number of senses for all common words.

It is important to emphasize here that, while we do use Wordnet in our work, our approach does not crucially rely on it, and we use it to obtain only very shallow information. In particular, we use Wordnet to obtain i) POS tags for Google 2-grams, and ii) number of senses for adjectives. POS tagging could be easily done using any one of the readily available POS taggers, but we chose this approach for its simplicity and speed. The number of different word senses for adjectives is harder to obtain without Wordnet, but this is only one of the four factors in our model, and we do not depend crucially on it.

5 Experiments

We evaluate our model in two stages. Firstly, using automatic evaluation with a set of jokes collected from Twitter, and secondly, by comparing our approach to human-generated jokes.

5.1 Inference

As the focus of this paper is on the model, not the inference methods, we use exact inference. While this is too expensive for estimating the true probability of any (x, y, z) triple, it is feasible if we fix one of the nouns, i.e., if we deal with $P(Y, Z|X = x)$. Note that this is only a limitation of our inference procedure, not the model, and future work will look at other ways (e.g., Gibbs sampling) to perform inference. However, generating Y and Z given X, such that the joke is funny, is still a formidable challenge that a lot of humans are not able to perform successfully (cf. performance of human-generated jokes in Table 2).

5.2 Automatic evaluation

In the automatic evaluation we measure the effect of the different factors in the model, as laid out in Section 3.1. We use two metrics for this evaluation. The first is similar to log-likelihood, i.e., the log of the probability that our model assigns to a triple. However, because we do not compute it on all the data, just on the data that contains the Xs from our development set, it is not exactly equal to the log-likelihood. It is a local approximation to log-likelihood, and we therefore dub it LOcal Log-likelihood, or LOL-likelihood for short. Our second metric computes the rank of the human-generated jokes in the distribution of all possible jokes sorted decreasingly by their LOL-likelihood. This Rank OF Likelihood (ROFL) is computed relative to the number of all possible jokes, and like LOL-likelihood is averaged over all the jokes in our development data. One advantage of ROFL is that it is designed with the way we generate jokes in mind (cf. Section 5.3), and thus more directly measures the quality of generated jokes than LOL-likelihood. For measuring LOL-likelihood and ROFL we use a set of 48 jokes randomly sampled from Twitter that fit the *I like my X like I like my Y, Z* pattern.

Table 1 shows the effect of the different factors on the two metrics. We use a model with only noun-attribute similarity (factors $\phi(X, Z)$ and $\phi(Y, Z)$) as the baseline. We see that the single biggest improvement comes from the attribute surprisal factor, i.e., from using rarer attributes. The best combination of the factors, according to automatic metrics, is using all factors except for the noun similarity (*Model 1*), while using all the factors is the second best combination (*Model 2*).

5.3 Human evaluation

The main evaluation of our model is in terms of human ratings, put simply: do humans find the jokes generated by our model funny? We compare four models: the two best models from Section 5.2

Model	LOL-likelihood	ROFL
Baseline	-225.3	0.1909
" + $\phi(X, Y)$	-227.1	0.2431
" + $\phi_1(Z)$	-204.9	0.1467
" + $\phi_2(Z)$	-224.6	0.1625
" + $\phi_1(Z)$ + $\phi_2(Z)$ (Model 1)	**-198.6**	**0.1002**
All factors (Model 2)	-203.7	0.1267

Table 1: Effect of different factors.

Model	κ	Mean	% of funny jokes
Human jokes	0.31	2.09	33.1
Baseline	0.58	2.78	3.7
Model 1	0.52	2.71	6.3
Model 2	0.58	2.56	16.3

Table 2: Comparison of different models on the task of generating Y and Z given X.

(one that uses all the factors (*Model 2*), and one that uses all factors except for the noun dissimilarity (*Model 1*)), a baseline model that uses only the noun-attribute similarity, and jokes generated by humans, collected from Twitter. We sample a further 32 jokes from Twitter, making sure that there was no overlap with the development set.

To generate a joke for a particular *x* we keep the top n most probable jokes according to the model, renormalize their probabilities so they sum to one, and sample from this reduced distribution. This allows our model to focus on the jokes that it considers "funny". In our experiments, we use $n = 30$, which ensures that we can still generate a variety of jokes for any given *x*.

In our experiments we showed five native English speakers the jokes from all the systems in a random, per rater, order. The raters were asked to score each joke on a 3-point Likert scale: 1 (funny), 2 (somewhat funny), and 3 (not funny). Naturally, the raters did not know which approach each joke was coming from. Our model was used to sample Y and Z variables, given the same Xs used in the jokes collected from Twitter.

Results are shown in Table 2. The second column shows the inter-rater agreement (Randolph, 2005), and we can see that it is generally good, but that it is lower on the set of human jokes. We inspected the human-generated jokes with high disagreement and found that the disagreement may be partly explained by raters missing cultural references in the jokes (e.g., a *sonic screwdriver* is Doctor Who's tool of choice, which might be lost on those who are not familiar with the show). We do not explicitly model cultural references, and are thus less likely to generate such jokes, leading to higher agreement. The third column shows the mean joke score (lower is better), and we can see that human-generated jokes were rated the funniest, jokes from the baseline model the least funny, and that the model which uses all the factors (*Model 2*) outperforms the model that was best according to the automatic evaluation (*Model 1*). Finally, the last column shows the percentage of jokes the raters scored as funny (i.e., the number of *funny* scores divided by the total number of scores). This is a metric that we are ultimately interested in – telling a joke that is somewhat funny is not useful, and we should only reward generating a joke that is found genuinely funny by humans. The last column shows that human-generated jokes are considered funnier than the machine-generated ones, but also that our model with all the factors does much better than the other two models. *Model 2* is significantly better than the baseline at $p = 0.05$ using a sign test, and human jokes are significantly better than all three models at $p = 0.05$ (because we were testing multiple hypotheses, we employed Holm-Bonferroni correction (Holm, 1979)). In the end, our best model generated jokes that were found funny by humans in 16% of cases, compared to 33% obtained by human-generated jokes.

Finally, we note that the funny jokes generated by our system are not simply repeats of the human jokes, but entirely new ones that we were not able to find anywhere online. Examples of the funny jokes generated by *Model 2* are shown in Table 3.

I like my relationships like I like my source, open
I like my coffee like I like my war, cold
I like my boys like I like my sectors, bad

Table 3: Example jokes generated by Model 2.

6 Conclusion

We have presented a fully unsupervised humor generation system for generating jokes of the type *I like my X like I like my Y, Z*, where X, Y, and Z are slots to be filled in. To the best of our knowledge, this is the first humor generation system that does not require any labeled data or hard-coded rules.

We express our assumptions about what makes a joke funny as a machine learning model and show that by estimating its parameters on large quantities of unlabeled data we can generate jokes that are found funny by humans. While our experiments show that human-generated jokes are funnier more of the time, our model significantly improves upon a non-trivial baseline, and we believe that the fact that humans found jokes generated by our model funny 16% of the time is encouraging.

Acknowledgements

The authors would like to thank the raters for their help and patience in labeling the (often not so funny) jokes. We would also like to thank Micha Elsner for this helpful comments. Finally, we thank the inhabitants of offices 3.48 and 3.38 for putting up with our sniggering every Friday afternoon.

References

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, AAAI '94, pages 633–638, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 107–116.

Christiane Fellbaum. 1998. *Wordnet: an electronic lexical database*. MIT Press.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Chloé Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies: short papers - Volume 2*, pages 89–94.

Igor Labutov and Hod Lipson. 2012. Humor as circuits in semantic networks. In *Proceedings of the 50th Annual Meeting of the ACL (Volume 2: Short Papers)*, pages 150–155, July.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Holberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2010. Quantitative analysis of culture using millions of digitized books. *Science*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: investigations in automatic humor recognition. In *Proceedings of the conference on Human Language Technology and EMNLP*, pages 531–538.

Justus J. Randolph. 2005. Free-marginal multirater kappa (multirater free): An alternative to fleiss fixed- marginal multirater kappa. In *Joensuu University Learning and Instruction Symposium*.

Jonas Sjöbergh and Kenji Araki. 2008. A complete and modestly funny system for generating and performing japanese stand-up comedy. In *Coling 2008: Companion volume: Posters*, pages 111–114, Manchester, UK, August. Coling 2008 Organizing Committee.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.