What Do You Mean I'm Funny? Personalizing the Joke Skill of a Voice-Controlled Virtual Assistant

Alejandro Mottini and Amber Roy Chowdhury
Amazon
Seattle
USA
(amottini,amberch)@amazon.com

Abstract

A considerable part of the success experienced by Voice-controlled virtual assistants (VVA) is due to the emotional and personalized experience they deliver, with humor being a key component in providing an engaging interaction. In this paper we describe methods used to improve the joke skill of a VVA through personalization. The first method, based on traditional NLP techniques, is robust and scalable. The others combine self-attentional network and multi-task learning to obtain better results, at the cost of added complexity. A significant challenge facing these systems is the lack of explicit user feedback needed to provide labels for the models. Instead, we explore the use of two implicit feedback-based labelling strategies. All models were evaluated on real production data. Online results show that models trained on any of the considered labels outperform a heuristic method, presenting a positive real-world impact on user satisfaction. Offline results suggest that the deep-learning approaches can improve the joke experience with respect to the other considered methods.

Introduction

Voice-controlled virtual assistants (VVA) such as Siri and Alexa have experienced an exponential growth in terms of number of users and provided capabilities. They are used by millions for a variety of tasks including shopping, playing music, and even telling jokes. Arguably, their success is due in part to the emotional and personalized experience they provide. One important aspect of this emotional interaction is humor, a fundamental element of communication. Not only can it create in the user a sense of personality, but also be used as fallback technique for out-of-domain queries (Bellegarda 2014). Usually, a VVA's humorous responses are invoked by users with the phrase *"Tell me a joke"*. In order to improve the joke experience and overall user satisfaction with a VVA, we propose to personalize the response to each request. To achieve this, a method should be able to recognize and evaluate humor, a challenging task that has been the focus of extensive work. Some authors have applied traditional NLP techniques (Yan and Pedersen 2017), while others deep learning models (Donahue, Romanov, and Rumshisky 2017). Moreover, (Yang et al. 2015) follows a semantic-based approach, while (Ruch 1992) and (Ahuja, Bali, and Singh 2018) tackle the challenge from a cognitive and linguistic perspective respectively.

To this end, we have developed two methods. The first one is based on traditional NLP techniques. Although relatively simple, it is robust, scalable, and has low latency, a fundamental property for real-time VVA systems. The other approaches combine multi-task learning (Caruana 1997) and self-attentional networks (Vaswani et al. 2017) to obtain better results, at the cost of added complexity. Both BERT (Devlin et al. 2018) and an adapted transformer (Vaswani et al. 2017) architecture are considered. This choice of architecture was motivated by the advantages it presents over traditional RNN and CNN models, including better performance (Liu et al. 2018), faster training/inference (important for real-time systems), and better sense disambiguation (Tang et al. 2018) (an important component of computational humor (Yang et al. 2015)).

The proposed models use binary classifiers to perform point-wise ranking, and therefore require a labelled dataset. To generate it, we explore two implicit user-feedback labelling strategies: five-minute reuse and one-day return. Online A/B testing is used to determine if these labelling strategies are suited to optimize the desired user-satisfaction metrics, and offline data to evaluated and compared the system's performance.

Method

Labelling Strategies

Generating labels for this VVA skill is challenging. Label generation through explicit user feedback is unavailable since asking users for feedback creates friction and degrade the user experience. In addition, available humor datasets such as (Yang et al. 2015; Potash, Romanov, and Rumshisky 2017) only contain jokes and corresponding labels, but not the additional features we need to personalize the jokes.

To overcome this difficulty, it is common to resort to implicit feedback. In particular, many VVA applications use interruptions as negative labels, the rationale being that unhappy users will stop the VVA. This strategy, however, is

Presented at the AAAI 2020 Workshop on Interactive and Conversational Recommendation Systems

Table 1: Example of labelling strategies: five-minute reuse (label 1) and 1-day return (label 2)

Timestamp	Label 1	Label 2
2019/05/03-17:51:10	1	0
2019/05/03-17:53:10	0	0
2019/05/06-21:41:09	1	1
2019/05/06-21:44:19	0	1
2019/05/07-20:34:19	0	0

Table 2: Examples of features within each category

Feature	Type	Category
Country Code	Categorical	User
Joke Text	String	Item
Request Timestamp	Timestamp	Context

not suitable for our use-case since responses are short and users need to hear the entire joke to decide if it is funny. Instead, we explore two other implicit feedback labelling strategies: five-minute reuse and 1-day return. Five-minute reuse labels an instance positive if it was followed by a new joke request within five-minutes. Conversely, 1-day return marks as positive all joke requests that were followed by a new one within the following 1 to 25-hour interval. Both strategies assume that if a user returns, he is happy with the jokes. This is clearly an approximation, since a returning user might be overall satisfied with the experience, but not with all the jokes. The same is true for the implied negatives; the user might have been satisfied with some or all of the jokes. Therefore, these labels are noisy and only provide weak supervision to the models.

Table 1 shows an example of the labels' values for a set of joke requests from one user.

Features

All models have access to the same raw features, which we conceptually separate into user, item and contextual features. Examples of features in each of these categories are shown in Table 2. Some of these are used directly by the models, while others need to be pre-processed. The manner in which each model consumes them is explained next.

NLP-based: LR-Model

To favor simplicity over accuracy, a logistic regression (LR) model is first proposed. Significant effort was put into finding expressive features. Categorical features are one-hot encoded and numerical ones are normalized. The raw Joke Text and Timestamp features require special treatment. The Joke Text is tokenized and the stop-words are removed. We can then compute computational humor features on the clean text such as sense combination (Yang et al. 2015) and ambiguity (Mihalcea and Pulman 2007). In addition, since many jokes in our corpus are related to specific events (Christmas, etc), we check for keywords that relate the jokes to them. For example, if "Santa" is included, we infer it is a Christmas joke. Finally, pre-computed word embeddings with sub-word information are used to represent jokes by taking the average and maximum vectors over the token representations. Sub-word information is important when encoding jokes since many can contain out-of-vocabulary tokens. The joke's vector representations are also used to compute a summarized view of the user's past liked and disliked jokes. We consider that a user liked a joke when the assigned label is 1, an approximation given the noisy nature of the labels. The user's liked/disliked joke vectors are also combined with the candidate joke vector by taking the cosine similarity between them.

For the raw Timestamp feature, we first extract simple time/date features such as month, day and isWeekend. We then compute binary features that mark if the timestamp occurred near one of the special events mentioned before. Some of these events occur the same day every year, while others change (for example, the Super Bowl). In addition, many events are country dependent. The timestamp's event features are combined with the joke's event features to allow the model to capture if an event-related joke occurs at the right time of the year.

The LR classifier is trained on the processed features and one of the labels. The model's posterior probability is used to sort the candidates, which are chosen randomly from a pool of unheard jokes. Although useful (see Validation section), this model has several shortcomings. In particular, many of the used features require significant feature engineering and/or are country/language dependent, limiting the extensibility of the model.

Deep-Learning-based: DL-Models

To overcome the LR-model's limitations, we propose the following model (see Figure 1). In the input layer, features are separated into context, item and user features. Unlike the LR-model, time and text features do not require extensive feature engineering. Instead, simple features (day, month and year) are extracted from the timestamp. After tokenization and stop-word removal, text features are passed through a pre-trained word embeding layer, and later, input into the joke encoder block. The basis of the joke encoder is a modified transformer. Firstly, only the encoder is needed. Moreover, since studies suggest that humor is subjective and conditioned on the user's context (Mulder and Nijholt 2002), we add an additional sub-layer in the transformer encoder that performs attention over the user's features. This sub-layer, inserted between the two typical transformer sub-layers at certain depths of the network, allows the encoder to adapt the representations of the jokes to different user contexts. Thus, the same joke can be encoded differently depending on the user's features. In practice, this additional sub-layer works like the normal self-attention sub-layer, except it creates its query matrix Q from the sub-layer below, and its K and V matrices from the user features. As an alternative, we also test encoding the jokes using a pre-trained BERT model.

Regardless of the used encoder, we average the token representations to obtain a global encoding of the jokes. The same encoder is used to represent the item's (the joke to

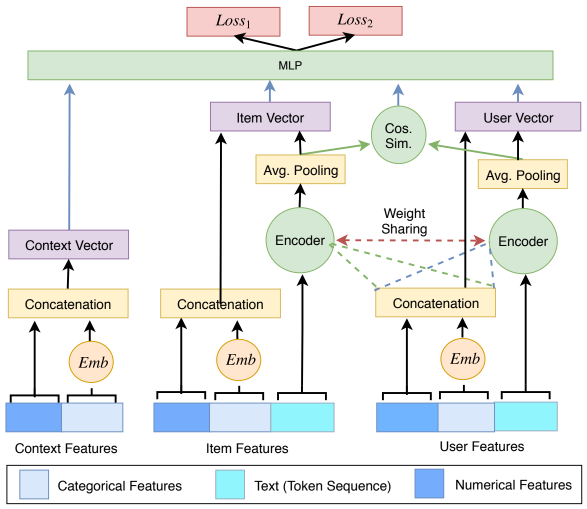

Figure 1: Architecture of the transformer-based model

rank) and the user's (liked and disliked jokes) textual features through weight sharing, and the cosine similarity between both representations are computed. The processed features are then concatenated and passed through a final block of fully connected layers that contains the output layers. Since experiments determined (see Validation section) that both labeling strategies can improve the desired business metrics, instead of optimizing for only one of them, we take a multi-task learning approach. Thus, we have two softmax outputs.

Finally, we use a loss function that considers label uncertainty, class imbalance and the different labeling functions. We start from the traditional cross-entropy loss for one labelling function. We then apply uniform label smoothing (Szegedy et al. 2016), which converts the one-hot-encoded label vectors into smoothed label vectors towards 0.5:

$$y_{ls} = y_{one-hot} * (1 - \epsilon) + \frac{\epsilon}{2} \tag{1}$$

with ϵ a hyper-parameter. Label smoothing provides a way of considering the uncertainty on the labels by encouraging the model to be less confident. We have also experimented with other alternatives, including specialized losses such as (Martinez and Stiefelhagen 2018). However, they did not produce a significant increase in performance in our tests. To further model the possible uncertainty in the feedback, we apply sample weights calculated using an exponential decay function on the time difference between the current and the following training instance of the same customer:

$$w_i = a * b^{t_i} + 1.0 \tag{2}$$

where w_i is the weight of sample i, t_i is the time difference between instances i and $i + 1$ for the same user, and a, b are hyper-parameters such that $a > 0$ and $0 < b < 1$. The rationale behind these weights is the following. If for example, we consider labeling function 1, and a user asks for consecutive jokes, first within 10 seconds and later within 4.9 minutes, both instances are labeled as positive. However, we hypothesize that there is a lower chance that in the second case the user requested an additional joke because he liked the first one. In addition, class weights are applied to each sample to account for the natural class imbalance of the dataset. Finally, the total loss to be optimized is the weighted sum of the losses for each of the considered labeling functions:

$$\mathcal{L}((f(x), \Theta), y) = \sum_{l=1}^{2} w_l \mathcal{L}_l \tag{3}$$

where w_l are manually set weights for each label and $\mathcal{L}_l$ are the losses corresponding to each label, which include all the weights mentioned before.

Validation

A two-step validation was conducted for English-speaking customers. An initial A/B testing for the LR model in a production setting was performed to compare the labelling strategies. A second offline comparison of the models was conducted on historical data and a selected labelling strategy. One month of data and a subset of the customers was used (approx. eighty thousand). The sampled dataset presents a fraction of positive labels of approximately 0.5 for reuse and 0.2 for one-day return. Importantly, since this evaluation is done on a subset of users, the dataset characteristic's do not necessarily represent real production traffic. The joke corpus in this dataset contains thousands of unique jokes of different categories (sci-fi, sports, etc) and types (puns, limerick, etc). The dataset was split timewise into training/validation/test sets, and hyperparameters were optimized to maximize the AUC-ROC on the validation set. As a benchmark, we also consider two additional methods: a non-personalized popularity model and one that follows (Kim 2014), replacing the transformer joke encoder with a CNN network (the specialized loss and other characteristics of the DL model are kept).

Hyperparameters were optimized using grid-search for the LR-Model. Due to computational constraints, random search was instead used for the DL-Model. In both cases, hyperparameters are selected to optimize the AUC-ROC on the validation set. Table 3 lists some of the considered hyperparameter values and ranges for both models. The actual optimal values are sample specific.

Online Results: A/B Testing

Two treatment groups are considered, one per label. Users in the control group are presented jokes at random, without repetition. Several user-satisfaction metrics such as user interruption rate, reuse of this and other VVA skills, and number of active dialogs are monitored during the tests. The relative improvement/decline of these metrics is compared between the treatments and control, and between the treatments themselves. The statistical significance is measured when determining differences between the groups. Results show that the LR-based model consistently outperforms the heuristic method for both labeling strategies, significantly improving retention, dialogs and interruptions. These results suggest that models trained using either label can improve the VVA's joke experience.

Table 3: Hyperparameter values tuned over, LR (top) and DL models (bottom)

Name	Value
Elastic-Net param.	[0.01,0.5]
Regularization	$[10^{-3},10.0]$
Fit intercept	True/False
Learning rate	$[10^{-3}, 10^{-5}]$
Batch size	[32,256]
Label smoothing	[0.1,0.3]
Keep probability	[0.5,0.8]
Num. heads	[2,6]
Num. transformer layers	[1,6]
F.C layers	[2,5]
F.C layer sizes	[16,256]
CNN filter sizes	[2,32]
CNN num. filters	[16,128]

Table 4: Relative change w.r.t popularity model of AUC-ROC and Overall Accuracy: transformer model (DL-T), BERT model (DL-BERT), transformer without special context-aware attention (DL-T-noAtt) and without both special attention and modified loss (DL-T-basic), CNN model (DL-CNN) and LR model (LR).

Method	R. Ch. AUC-ROC	R. Ch. O.A.
DL-T	0.31	0.24
DL-BERT	0.30	0.27
DL-T-noAtt	0.29	0.24
DL-T-basic	0.28	0.23
DL-CNN	0.28	0.13
LR	0.24	0.21

Offline Results

One-day return was selected for the offline evaluation because models trained on it have a better AUC-ROC, and both labeling strategies were successful in the online validation. All results are expressed as relative change with respect to the popularity model.

We start by evaluating the models using AUC-ROC. As seen in Table 4, the transformer-based models, and in particular our custom architecture, outperform all other approaches. Similar conclusions can be reached regarding overall accuracy. However, given the class imbalance, accuracy is not necessarily the best metric to consider. In addition, to better understand the effect to the original transformer architecture, we present the performance of the model with and without the modified loss and special attention sub-layer (see Table 4). Results suggest both modifications have a positive impact on the performance. Finally, to further evaluate the ranking capabilities of the proposed methods, we use top-1 accuracy. Additional positions in the ranking are not considered because only the top ranked joke is presented to the customer. Results show that the DL based models outperform the other systems, with a relative change in top-1 accuracy of 1.4 for DL-BERT and 0.43 for DL-T, compared with 0.14 for the LR method.

Results show that the proposed methods provide different compromises between accuracy, scalability and robustness. On one hand, the relatively good performance of the LR model with engineered features provides a strong baseline both in terms of accuracy and training/inference performance, at the cost of being difficult to extend to new countries and languages. On the other hand, DL based methods give a significant accuracy gain and require no feature engineering, which facilitates the expansion of the joke experience to new markets and languages. This comes at a cost of added complexity if deployed in production. In addition, given the size of the BERT model (340M parameters), real-time inference using DL-BERT becomes problematic due to latency constraints. In this regard, the DL-T model could be a good compromise since its complexity can be adapted, and it provides good overall accuracy.

Conclusions and Future Work

This paper describes systems to personalize a VVA's joke experience using NLP and deep-learning techniques that provide different compromises between accuracy, scalability and robustness. Implicit feedback signals are used to generate weak labels and provide supervision to the ranking models. Results on production data show that models trained on any of the considered labels present a positive real-world impact on user satisfaction, and that the deep learning approaches can potentially improve the joke skill with respect to the other considered methods. In the future, we would like to compare all methods in A/B testing, and to extend the models to other languages.

References

Ahuja, V.; Bali, T.; and Singh, N. 2018. What makes us laugh? investigations into automatic humor classification. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, 1–9.

Bellegarda, J. R. 2014. Spoken language understanding for natural interaction: The siri experience. *Natural Interaction with Robots, Knowbots and Smartphones.*

Caruana, R. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Donahue, D.; Romanov, A.; and Rumshisky, A. 2017. Humorhawk at semeval-2017 task 6: Mixing meaning and sound for humor recognition. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 98–102.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882.*

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198.*

Martinez, M., and Stiefelhagen, R. 2018. Taming the cross entropy loss. *arXiv preprint arXiv:1810.05075*.

Mihalcea, R., and Pulman, S. 2007. Characterizing humour: An exploration of features in humorous texts. In *International Conference on Intelligent Text Processing and Computational Linguistics*, 337–347. Springer.

Mulder, M., and Nijholt, A. 2002. *Humour Research: State of Art*, volume 02 of *CTIT Technical Report Series*. Netherlands: Centre for Telematics and Information Technology (CTIT). Imported from CTIT.

Potash, P.; Romanov, A.; and Rumshisky, A. 2017. Semeval-2017 task 6:# hashtagwars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 49–57.

Ruch, W. 1992. Assessment of appreciation of humor: Studies with the 3 wd humor test. *Advances in personality assessment* 9:27–75.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Tang, G.; Müller, M.; Rios, A.; and Sennrich, R. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Yan, X., and Pedersen, T. 2017. Who's to say what's funny? a computer using language models and deep learning, that's who! *arXiv preprint arXiv:1705.10272*.

Yang, D.; Lavie, A.; Dyer, C.; and Hovy, E. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2367–2376.