Sentiment Analysis with Eight Dimensions for Emotional Chatbots

Xianchao Wu[+], Yuichiro Kikura[*], Momo Klyen[+], Zhan Chen[+]
[+] Microsoft Development Co., Ltd
Shinagawa Grand Central Tower, 2-16-3 Konan Minato-ku, Tokyo 108-0075
{xiancwu, momokl, zhanc}@microsoft.com
[*] Graduate School of Information Science and Technology, The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656
kikura@mi.t.u-tokyo.ac.jp

1 Introduction

Understanding million level users' psychological emotions through machine learning techniques remains as a fundamental challenge for developing open domain free chatting oriented emotional chatbots, such as Rinna (Wu et al. 2016), a chat-oriented artificial intelligence (AI) character who is designed to be a senior high school girl. The major obstacles that this paper tries to deal with include:

(1) Existing benchmark data sets with three emotional categories of "positive", "negative", "neutral" or further with "strong positive" or "strong negative" are *deficient* to describe real application scenarios of chatting with chatbots. The difficulties include how to define the emotion taxonomy to better cover people's dominant sentiment feelings and consequently how to prepare a large-scale training data making use of the defined emotion category taxonomy;

(2) Spoken languages are mainly used during users' conversations with chatbots. Ambiguous boundaries of emotional words reduce the final accuracy of sentiment analysis (SA) models. However, it is not trivial for building a word segmentation model for spoken languages such as Japanese and Chinese to cover the wildly used abbreviations, emoji/kaomoji, and informal words;

(3) Speech and facial images play also very important roles for emotion expressing and transferring during real human's conversations. It will be interesting to simultaneously consider emotional signals from voices and facial images when building a text-oriented SA model.

In this paper, we borrow the emotion taxonomy from the emotion API for classifying facial image, which is a part of Microsoft's cognitive service[1]. In the taxonomy, eight dimensions are used to describe facial images' fine-grained emotions:

1. "happiness": "喜び", for example, "好きだからりんなさん"/Rinna, that is because I like you, "俺は可愛いと信じてる!"/I believe that I am cute.

2. "surprise":"驚き", for example, "台風がすごい"/The typhoon is shocking, "うぇ!?ほんとですか?"/what?! Really?

3. "anger": "怒り", for example, "お前無視すんなよ"/How dare you ignore that, "面白くないは、(怒る)"/it's not interesting, (angry).

4. "disgust":"嫌悪", for example, "別にお前に嫌われたっていいし" /I do not care that I am disgusted by you, "思ったより君頭悪いね"/You are more stupid than I expected.

5. "sadness": "悲しみ", for example, "いやだ。泣きたい。"/it's disgusting and I am feeling crying, "毎日が悲しくなる"/I am feeling sadder every day.

6. "contempt": "軽蔑", for example, "AI ちょっと軽蔑してるよ"/AI is despising me, "コンピュータのくせに, 威張ってんじゃねーぞ"/only a computer cannot be that swagger.

7. "fear": "恐怖", for example, "今から? 怖い番組があるで?"/from now on, there will be a scary TV program!?, "怖い話 10 回続けて言って"/say scary sentences 10 times.

8. "neutral": "中性", for example, "明日のスケジュールが決めました"/Tomorrow's schedule is determined, "来週の東京の天気を知りたいです"/I want to know next week's weather of Tokyo.

Through borrowing this emotion category set from facial image classification, we hope to build a bridge between SA of facial images as well as texts which further have a deep connection with speech. Another reason of using this category set is that our emotional chatbot is intended to take care of users' detailed *negative* emotions rather than positive emotions. In the 8 labels, "happiness" is the only positive emotion and there are six types of negative or relatively negative emotions except "neutral".

After determining the y set in our SA model, we collect large-scale training data in the form of $<x, y>$ starting from seed emotional lexicons (with emoji/kaomoji and emotional words included) and seed sentences. Each x here is a sentence that includes a sequence of characters. The details will be described in Section 2.

[*] Work done when Kikura was an internship student in Microsoft.
[1] https://www.microsoft.com/cognitive-services/en-us/emotion-api

Then, we tackle the word segmentation obstacle by adopting a Recurrent Convolutional Neural Network (RCNN) proposed by Kim et al. (2016) that directly takes sequences of Japanese characters as inputs and then use a convolutional function to automatically extract the n-gram characters as representations of word/phrase features. We further use a recurrent layer that accepts the features from the convolutional layer and embed the whole sentence in a character order sensitive way. The output layer is a softmax layer that maps dense vector representations from the recurrent layer to the eight emotional dimensions. The details will be described in Section 3.

We finally express our experiments and illustrate the usage of the SA model to a real-word emotional chatbot, Rinna, who is communicating with more than five million friends.

2 Training Data Collection

We depict our training data collection pipeline in Figure 1. In order to construct a large-scale <text, emotion category> training data, we make use of two seeds to obtain large-scale training data.

First, we expand seed emotional words by using word2vec (Mikolov et al. 2013) and bilingual word alignment table (Brown et al. 1993). Using word2vec, we can obtain a high similarity score for two words that share quite similar context information. However, one problem is that words such as "black" and "white" will have a relatively high similarity score since they both are adjective and are used to modify the color of an object. We thus further make sue of bilingual word alignment table for further collecting and pruning the expanded seed emotional words.

Second, we manually collect emoji/kaomoji of these eight emotion categories from the Web and then append these emoji/kaomoji into the seed lexicon.

In Figure 1, an example of taking a seed word "悲しみ/sadness" for word2vec word extension is shown. The cosine function similarity scores are also computed. For example, the word "悲しみ/sadness" and "哀しみ/sorrow" has a cosine similarity score of 0.69 (the larger the score, the closer their semantic meanings). As former mentioned, word2vec is not guarantee that the result words are share a "similar" meaning with the seed word of "悲しみ/sadness", such as bad cases of "永遠/forever" and "喜び/happiness". To alleviate this problem, we leverage bilingual word alignment table to remove these bad cases. That is, after this word2vec word extension, we further make use of Japanese-to-English word alignment table for finding English words that are aligned with "悲しみ/sadness". For example, we found four words, which are "sad", "unhappy", "sorrowful", and "pathetic". Then, we use the word alignment table in another direction of from English to Japanese to obtain Japanese words for each English word. All these Japanese words will form another word list. We finally use an "interaction" operation to the word list from word2vec and from word-alignment. The result word list will be appended to "Seed word lexicon".

At the same time, we manually collect emoji/kaomoji from the web and then append them to the "seed word lexicon" as well. Emoji and Kaomoji examples of the eight categories are illustrated in Figure 1 as well. The result "seed word lexicon" will be used to find sentences that contain at least one seed word in the web data. We can obtain a large-scale training data in the form of <text, emotion category> through this way. Consequently, we can use of the yielded emotional lexicon and training data to build <voice, emotion category> for the final task of voice emotion classification.

However, it is risky to use maximum length matching style methods to collect the final large-scale training data using the seed word lexicon. For one reason is about the "not"-series words which switch the original emotion into a contrary direction. For another reason is

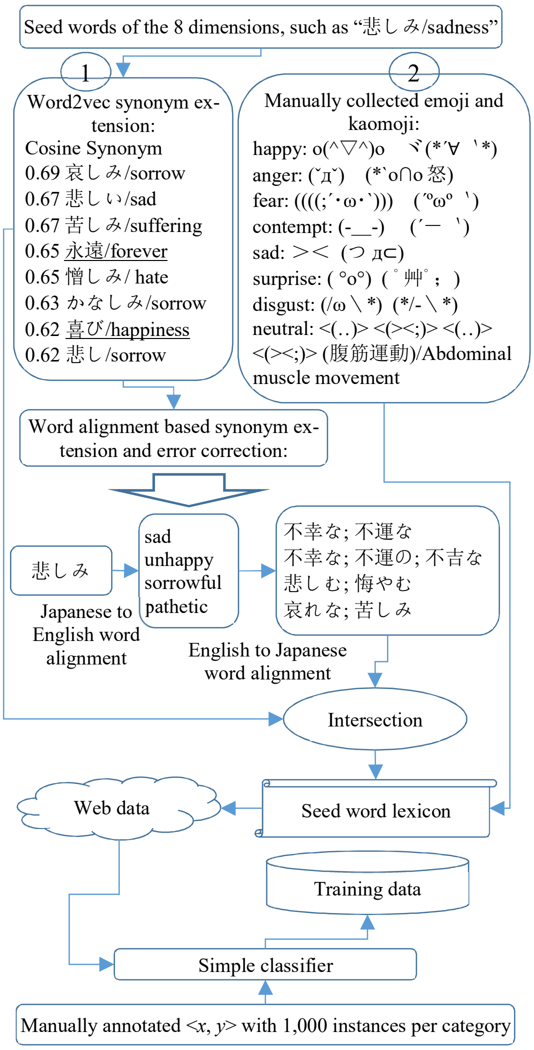

Figure 1. Pipeline for training data collection.

that one sentence can contain both positive words and negative words in a mixture way such as “praise first and then criticize” or “criticize first and then praise”. In order to alleviate these problems, we manually annotate a seed training data with 1,000 instances per category. For the “neutral” category we do not annotate it since the instances can be easily yielded by collecting the sentences that do not have any emotional words or emoji/kaomoji inside it.

We consequently train a simple classifier that utilizes n-gram character language model features. The classifier make a secondary judgement to the web data pre-filtered by the seed word lexicon. The sentences that have a relatively high confidence probability will be finally appended to our training data set (also refer to the bottom side of Figure 1).

3 Character-level RCNN

The character-level RCNN language models (Kim et al. 2016) were verified to be able to encode, from characters only, both semantic and orthographic information. Figure 2 depicts the architecture overview in which we customized the structure for our task’s usage. First, each character in sentence are converted into dense vector spaces alike bag of words neural language models. Next, convolution neural network (CNN) initially described in (LeCun, 1989) converts them with various kernel sizes. Then the vectors are transferred to the recurrent neural network (RNN) layer in which long-short term memory (LSTM) units are employed. Finally, aiming at solving the problem described in this paper, the states of RNN are regarded as feature vectors and are passed to the softmax layer for multiple category emotion classification.

Note that the major merit of the architecture is that the recurrent layer takes the output from a single-layer character-level convolutional neural network with max-over-time pooling as input. LSTM (Hochreiter and Schmidhuber 1997) addresses (1) the learning of long distance dependencies and (2) the gradient vanishing problem by augmenting the traditional RNN with a memory cell vector $\mathbf{c}_t \in \mathbb{R}^n$ at each time step. Formally, one step of an LSTM takes as input $\mathbf{x}_t$, $\mathbf{h}_{t\text{-}1}$, $\mathbf{c}_{t\text{-}1}$ and produces $\mathbf{h}_t$, $\mathbf{c}_t$ via the following intermediate calculations:

$$\mathbf{i}_t = \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t\text{-}1} + \mathbf{b}^i),$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t\text{-}1} + \mathbf{b}^f),$$
$$\mathbf{o}_t = \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t\text{-}1} + \mathbf{b}^o),$$
$$\mathbf{g}_t = \tanh(\mathbf{W}^g\mathbf{x}_t + \mathbf{U}^g\mathbf{h}_{t\text{-}1} + \mathbf{b}^g),$$
$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t\text{-}1} + \mathbf{i}_t \otimes \mathbf{g}_t,$$
$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t).$$

Here σ(.) and tanh(.) are the element-wise sigmoid and hyperbolic tangent functions, ⊗ is the element-wise multiplication operator, and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ respectively denote *input*, *forget*, and *output* gates. When t = 1, $\mathbf{h}_0$ and $\mathbf{c}_0$ are initialized to be zero vectors. Parameters to be trained of the LSTM layer are matrices $\mathbf{W}^j$, $\mathbf{U}^j$, and the bias vector $\mathbf{b}^j$ for $j \in \{i, f, o, g\}$.

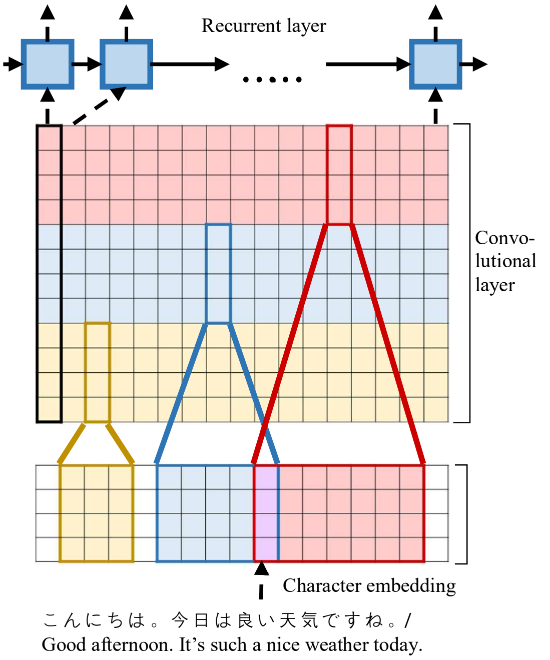

Figure 2. Architecture of our character-level RCNN with three major layers drawn.

CNNs have achieved state-of-the-art results on computer vision tasks such as the ImageNet shared tasks and have also shown to be effective for various NLP tasks (Collobert et al. 2011). Since NLP tasks’ inputs are one dimension word orders instead of 2D images, the CNN architectures employed for NLP applications differ in that they typically involve temporal rather than spatial convolution functions. Let $\mathbf{Q} \in \mathbb{R}^{d*|V|}$ be the character embedding matrix with d being the dimensionality of character embedding and V being the character vocabulary set. Suppose that word $w = c_1, \ldots, c_l$ with l characters. Then, the character-level representation of w is given by a matrix $\mathbf{C}^w \in \mathbb{R}^{d*l}$, where the j-th column corresponds to the character embedding for c_j which is further the c_j-th column of $\mathbf{Q}$. We apply a narrow convolution between $\mathbf{C}^w$ and a *filter* (or convolutional function) $\mathbf{H} \in \mathbb{R}^{d*f}$ of width f (Figure 2 shows examples of f = 3, 5, 7 and we further used f = 9 in our experiments), after which we add a bias and then apply a nonlinearity to obtain a feature map $\mathbf{f}^w \in \mathbb{R}^{l\text{-}f+1}$. Specifically, the i-th element of $\mathbf{f}^w$ is given by:

$$\mathbf{f}^w[i] = \tanh(\langle \mathbf{C}^w[*, i{:}i{+}f\text{-}1], \mathbf{H}\rangle + b),$$

where $\mathbf{C}^w[*, i{:}i{+}f\text{-}1]$ is the i-to-$(i{+}f\text{-}1)$-th column of $\mathbf{C}^w$ and $\langle \mathbf{A}, \mathbf{B}\rangle = \mathrm{Tr}(\mathbf{AB}^T)$ is the Frobenius inner product. Finally, we take the *max-over-time* pooling result,

$$y^w = \max_i \mathbf{f}^w[i]$$

as the feature corresponding to the filter $\mathbf{H}$ (when applied to word w). The idea behand is to capture the most “important” feature (i.e., a character n-gram) where the size of the feature corresponds to the filter width f. Suppose we have a total of h filters $\mathbf{H}_1, \ldots, \mathbf{H}_h$, then $\mathbf{y}^w = [y_1^w, \ldots, y_h^w]$ is the representation of word w.

4 Experiments

Category	Number	Ratio	AvgLen
Happiness 喜び	956,007	56.2%	24.9
Surprise 驚き	42,322	2.5%	26.2
Anger 怒り	51,065	3.0%	27.1
Disgust 嫌悪	4,748	0.3%	23.9
Sadness 悲しみ	562,945	33.1%	25.7
Contempt 軽蔑	42,775	2.5%	25.1
Fear 恐怖	41,039	2.4%	31.7

Table 1. Statistical Information of the training data. AvgLen stands for the average character number per sentence.

We use 1.5-year Japanese twitter data as the "Web data" (Figure 1). For training a word2vec model with 200 dimensions, we use the Japanese Wikipedia and Bing's large-scale queries. The total data is 80GB with a vocabulary size of 7.3 million. We used an in-house CRF-style word segmentation model. We manually collected more than 1,000 emoji/kaomoji for each of the 7 categories except the "neutral" category. The statistical information of the final training data is given in Table 1. Note that the positive category "happiness" takes a share of 56.2% which is larger than the ratio of all the other six categories. We further randomly sample a "neutral" category data from the rest of the "Web data" with a size of 1 million sentences. We take a 9:0.5:0.5 separating of the data for training/validating/testing. Table 2 shows that Char-RCNN model is significantly better (+4.0%, +7.9%) than two baseline

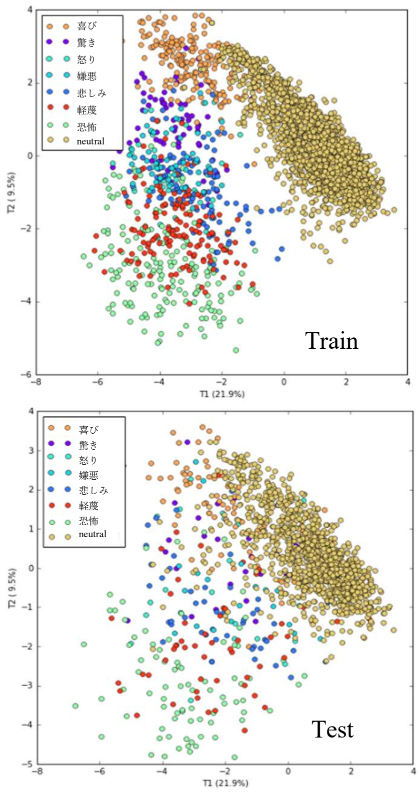

Figure 3. Feature space visualization by PCA.

Models	Accuracy
n-gram (n = 3) + SVM	0.844
Word2vec + RNN + softmax	0.805
Char-RCNN	**0.884**

Table 2. SA accuracy Comparison.

word-level systems. We also find that the n-gram features under SVM performs better than a vanilla RNN model taking word2vec embedding matrix as input and softmax as the output layer. Figure 3 depicts Char-RCNN's feature space visualization by PCA. In the two figures, "neutral" and other categories are separated into two major groups. The interesting part is that there is a gradually vertical distribution from "happiness", to "surprise", "anger", "disgust", "sadness", "contempt" and finally to "fear". Meanwhile, "happiness" is close to "surprise" yet far from "fear". Finally, we replace the 3-category SA model by this model in our ranker of Rinna, trained using gradient boosting decision trees (Wu et al. 2016). The accuracy improves from 76.0% to 76.7% which is especially effective for the emotional portion (20%) of the ranker's test data.

5 Conclusion

We have proposed an eight-dimension oriented SA system for emotional chatbots. We described the pipeline of large-scale training data collection and the architecture of the character RCNN classifier adopted from character-level RCNN language models (Kim et al. 2016). Experimental results shew that our SA model significantly outcomes the word-based SA models which suffer from Japanese word segmentation problem of spoken language. In the future, it will be interesting to investigate the combined training of joint SA models for both facial images and texts using training data such as movie frames with subtitles.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. Computational Linguistics.

K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," EMNLP 2014.

Collobert, R.;Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural Language Processing (almost) from Scratch. Journal of Machine Learning Research 12:2493–2537.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. Neural Computation 9:1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In AAAI 2016.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Handwritten Digit Recognition with a Backpropagation Network. In NIPS.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In NIPS 2013.

Xianchao Wu, Kazushige Ito, Katsuya Iida, Kazuna Tsuboi, Momo Klyen. りんな：女子高生人工知能. 言語処理学会 2016.