Supplementary Material: Pretraining Sentiment Classifiers with Unlabeled Dialog Data

Toru Shimizu[1], **Hayato Kobayashi**[1,2], **and Nobuyuki Shimizu**[1]

[1]Yahoo Japan Corporation
[2]Riken AIP
{toshimiz,hakobaya,nobushim}@yahoo-corp.jp

A Details of Sentiment Analysis Service

Figure 1 shows a screenshot of the tweet search/analysis service in the company that provided us the datasets. The service retrieves Twitter's tweets for given queries from a search index consisting of the Japanese portion of the Firehose data and presents them in a timeline manner. In addition to the timeline, as in the sentiment widget at the bottom right of the screenshot, it provides information with regard to sentiment over the query word, i.e., aggregating sentiment labels of the retrieved tweets into ratios. The green right part of the pie chart represents the percentage of positive sentiment, and the red left part corresponds to negative sentiment. The chart at the bottom shows how these ratios change depending on time.

What a user wants to know by searching on this service is, in many cases, what other people are thinking or feeling about a topic word (e.g. a musician, pop star, TV program, sports match, natural disaster, or current event), and this functionality serves the "feeling" part of user needs. To prepare sentiment labels in advance, a sentiment classifier resides in the backend system built on Apache Kafka[1] and Apache Storm[2] and labels each incoming tweet as either `positive`, `negative`, or `neutral` before it is indexed. Therefore, for this system, the message-level sentiment-analysis performance needs to be improved to make the aggregated percentages of the labels more accurate.

B Details of Proposed Method

Our approach consists of two steps: RNN pretraining and sentiment classification. We describe the RNN encoder-decoder dialog model, which is essential to the pretraining step, in Section B.1 and describe our contribution in Section B.2.

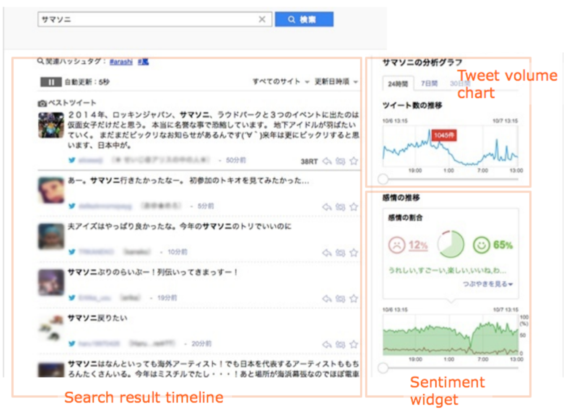
[1]`https://kafka.apache.org`
[2]`https://storm.apache.org`

B.1 Dialog Model

We first train an RNN encoder-decoder dialog model (Sutskever et al., 2014) using unlabeled conversational data, e.g., from Twitter, including a large number of tweet (source tweet) and reply (target tweet) pairs. The resultant dialog model can make a prediction about what is likely to come to the target side conditioned on a source tweet. In this section, we use the term "tweet" to make the explanation more precise even though our method is also applicable to other types of conversational pairs of conversational posts.

Figures 2(a), (c), and (d) illustrate the typical structures of RNN encoder-decoder dialog models. The encoder RNN consumes a source tweet and produces its vector representation, and the decoder RNN makes a prediction over the target side on the basis of the encoder's output.

Now let us assume that we are evaluating how appropriate a target tweet $\boldsymbol{v}$ is according to an RNN encoder-decoder model given a corresponding source tweet $\boldsymbol{u}$, i.e., computing $p(\boldsymbol{v}|\boldsymbol{u})$. For the preprocessing, we encode tweets into se-



２０１４年、ロッキンジャパン、サマソニ、ラウドパークと３つのイベントに出たのは仮面女子だけだと思う。本当に名誉な事で恐縮しています。地下アイドルが羽ばたいていく。まだまだビックリなお知らせがあるんです('∀｀)来年は更にビックリすると思います、日本中が。
50分前 38RT

あー。サマソニ行きたかったなー。初参加のトキオを見てみたかった…
5分前

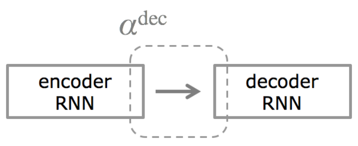
夫アイズはやっぱり良かったな。今年のサマソニのトリでいいのに
10分前

サマソニぶりのらいぶー！列伝いってきまっすー！
19分前

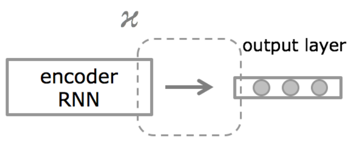
サマソニ戻りたい
20分前

サマソニはなんといっても海外アーティスト！でも日本を代表するアーティストももちろんたくさんいる。今年はミスチルでたし・・・！あと場所が海浜幕張なのでほぼ電車

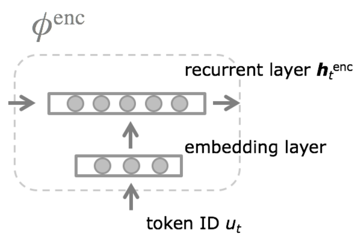
Figure 1: Result page of the tweet search/analysis service



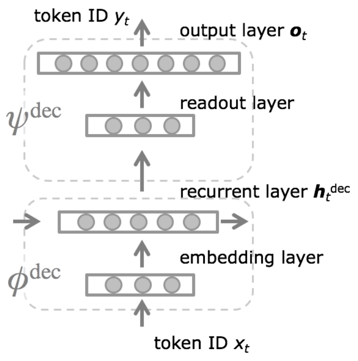
(a) Encoder-decoder model



(b) Sentiment-classifier model



(c) Encoder RNN



(d) Decoder RNN

Figure 2: Model overview

quences of token IDs that belong to a vocabulary of size K, $V = \{1, \ldots, K\}$; thus, mapping $\boldsymbol{u}$ into $(u_1, \ldots, u_{T_u})$ and $\boldsymbol{v}$ into $(v_1, \ldots, v_{T_v})$. The vocabulary includes a NULL token representing the start and end of the tweet.

First, the encoder RNN with a transition function ϕ^{enc} reads a source tweet $\boldsymbol{u}$ recurrently, i.e., $\boldsymbol{h}_t^{\text{enc}} = \phi^{\text{enc}}(\boldsymbol{h}_{t-1}^{\text{enc}}, u_t)$, as shown in Figure 2(c). Its hidden vector at the last time step $\boldsymbol{h}_{T_u}^{\text{enc}}$ reflects the whole sequence. By converting it into a vector representation with the function α^{dec} in Figure 2(a), we obtain an initial value for the decoder's hidden layer: $\boldsymbol{h}_0^{\text{dec}} = \alpha^{\text{dec}}(\boldsymbol{h}_{T_u}^{\text{enc}})$. After that, we map the target tweet $\boldsymbol{v}$ to an input sequence $\boldsymbol{x} = (\texttt{NULL}, v_1, \ldots, v_{T_v-1}, v_{T_v})$ and output sequence $\boldsymbol{y} = (v_1, v_2, \ldots, v_{T_v}, \texttt{NULL})$, aligning their elements so as to have the decoder predict the next tokens individually at each time step. The decoder RNN $\{\phi^{\text{dec}}, \psi^{\text{dec}}\}$ in Figure 2(d) reads $\boldsymbol{x}$ and predicts $\boldsymbol{y}$ recurrently, i.e., $\boldsymbol{h}_t^{\text{dec}} = \phi^{\text{dec}}(\boldsymbol{h}_{t-1}^{\text{dec}}, x_t)$, using $\boldsymbol{h}_0^{\text{dec}}$ as the initial hidden vector. To predict an output token y_t, we apply the output function ψ^{dec} to the hidden layer vector:

$$\boldsymbol{o}_t = \psi^{\text{dec}}(\boldsymbol{h}_t^{\text{dec}}), \tag{1}$$

$$p(y_t = i | \boldsymbol{h}_t^{\text{dec}}) = [\boldsymbol{o}_t]_i \,, \tag{2}$$

where $[\cdot]_i$ denotes the i-th element of a vector. Considering that the decoder hidden layer vector $\boldsymbol{h}_t^{\text{dec}}$ reflects the context $\boldsymbol{u}, y_1, \ldots, y_{t-1}$, we can obtain a probabilistic model for the source-target pair $\{\boldsymbol{u}, \boldsymbol{v}\}$ as

$$p(\boldsymbol{v}|\boldsymbol{u}) = \prod_{t=1}^{T} p(y_t|\boldsymbol{u}, y_1, \ldots, y_{t-1}) = \prod_{t=1}^{T} [\boldsymbol{o}_t]_{y_t} \,, \tag{3}$$

where T is set as $T_v + 1$. When we train this model over a collection of pairs $\{(\boldsymbol{u}^{(1)}, \boldsymbol{v}^{(1)}), \ldots, (\boldsymbol{u}^{(N)}, \boldsymbol{v}^{(N)})\}$, the cross-entropy cost function is

$$-\sum_{k=1}^{N} \log p(\boldsymbol{v}^{(k)}|\boldsymbol{u}^{(k)}). \tag{4}$$

Now we can train the encoder-decoder model $\{\phi^{\text{enc}}, \alpha^{\text{dec}}, \phi^{\text{dec}}, \psi^{\text{dec}}\}$ to minimize cost.

B.2 Connecting to Sentiment Classifier from Dialog Model

We train the encoder-decoder model with large-scale conversational data as in Section B.1. To exploit the learned ability, we take out parameters from the encoder RNN of the dialog model ϕ^{enc} and reuse them in the RNN sentiment analysis models in Figure 2(b).

For conducting sentiment analysis, we use only the encoder part of the encoder-decoder and join it with another network κ, which consists of one fully connected layer and softmax nonlinearity in the simplest settings. We apply the joined functions to an input tweet to analyze the sentiment. The encoder ϕ^{enc} generates a vector representation $\boldsymbol{h}_{T_u}^{\text{enc}}$, and κ maps the representation to a probability distribution over sentiment classes `positive`, `negative`, and `neutral` for the input, as in Figure 2(b). We call this combined model a sentiment classifier. We train the network $\{\phi^{\text{enc}}, \kappa\}$ in a supervised manner, consuming a set of tweets accompanied with sentiment labels.

C Examples of Generated Replies

Table 1 shows generated replies based on the pretrained encoder-decoder model. In these instances, the sentiment on the source side is well reflected on the target side. This suggests that the dialog model has learned a way to read out sentiment from a given source tweet.

D Parameter Tuning of LIBLINEAR

The following commands were used for parameter tuning of LIBLINEAR.

- `LogReg`: "train -s0 -ex", s.t. $x \in \{0.01, 0.1, 1, 2, 4\}$.
- `LinSVM`: "train -s2 -ex -py", s.t. $x \in \{0.01, 0.1, 1, 2, 4\}$, $y \in \{0, 0.01, 0.1, 1, 2, 4\}$.

E Details of Emoticon-based Data

The 2M dataset was created from 44.6M tweets in the training set of our dialog data. Since the 2M dataset is much smaller than the original dialog data, we prepared the 6M dataset additionally using another 92M tweets. We included each tweet as two instances: removing the emoticons from the one and leaving them in the other. In this way, we can train models that can make predictions on the basis of not only text without emoticons but also emoticons themselves as features, as in (Go et al., 2009).

F Accuracy of Sentiment Classification

Table 2 shows the accuracy results of the compared models on the sentiment classification task when varying data size (5K to 80K). Each value is the average of five trials with different random seeds for each setting.

References

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. Technical report, Stanford Digital Library Technologies Project. https://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Curran Associates, Inc., pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Original text		English translations	
Source tweet	Generated reply	Source tweet	Generated reply
明日は待ちに待ったコンサートだよ	いいね！	Tomorrow I have a concert I've been really looking forward to	That's nice!
私もっ♪ (´∀｀) 人 (´∀｀) ♪	(*^^*)	Me too ♪ (high five emoticon) ♪	:)
残念だったね	(´・ω・｀)	I'm sorry to hear that	:(
後でそっちに行くよ	おっけー！	I'm coming later	OK!
頭痛いよ	うそ、、 お大事に…	I have a headache	Really? Take care of your-self...
アメトーク見たかった〜	おもろいよね〜	I missed Ame Talk (a TV program)	Watching it is fun
もう、ごめんじゃ済まされないだろ、呆れる	それはそれで困る。。	Sorry doesn't cut it anymore. I gave up on you.	That's too bad...
大学 合格したよ！	おめでとう！！	I was admitted by the university!	Congratula-tions!!
もうだめだ	そんなことないよ (´・ω・‘)	It's all over for me	I don't think so :(
嘘つきめ。	ひどい	You liar.	You nasty
ちょうどいいね	まじかー ありがとう！！！	That's just right	Really? Thanks!!!
それ、すごい好き	うん、かっこいいよね	I really like it	Yeah, it's so cool

Table 1: Replies generated by pretrained encoder-decoder model

	5K	10K	20K	40K	80K
`Default`	68.47	71.48	72.86	75.07	76.50
`Dial`	**75.57**†	**76.79**†	**77.84**†	**78.80**†	**80.04**†
`Lang`	74.49	75.51	76.80	78.04	79.26
`SeqAE`	70.53	72.34	73.45	75.18	76.46
`Emo2M`	67.71	68.88	70.47	73.08	75.75
`Emo6M`	67.79	68.47	70.42	72.72	74.86
`LogReg`	70.87	71.93	73.49	74.59	75.80
`LinSVM`	70.25	71.67	73.11	73.75	74.20

Table 2: Accuracy (%) of sentiment classification of each model versus labeled data size. `Dial` is our proposed method, and † in its row indicates statistically significant difference from the corresponding value of `Lang` ($p < 0.05$).