RSL17BD at DBDC3: Computing Utterance Similarities based on Term Frequency and Word Embedding Vectors

Sosuke Kato[1], *Tetsuya Sakai*[1]

[1]Waseda University, Japan

sow@suou.waseda.jp, tetsuyasakai@acm.org

Abstract

RSL17BD (Waseda University Sakai Laboratory) participated in the Third Dialogue Breakdown Detection Challenge (DBDC3) and submitted three runs to both English and Japanese subtasks. Following the approach of Sugiyama, we utilise ExtraTreesRegressor, but instead of his simple word overlap feature, we employ term frequency vectors and word embedding vectors to compute utterance similarities. Given a target system utterance, we use ExtraTreesRegressor to estimate the mean and variance of its breakdown probability distribution, and then derive the breakdown probabilities from them. To calculate word embedding vector similarities between two neighbouring utterances, Run 1 follows the approach of Omari et al. and uses the maximum cosine similarity and geometric mean; Run 3 uses arithmetic mean instead; Run 2 utilises the cosine similarities of all term pairs from the two utterances. Run 2 statistically significantly outperforms the other two for the English data ($p = 0.011$ with Jensen-Shannon Divergence and $p = 0.009$ with Mean Squared Error).

Index Terms: dialogue breakdown detection, ExtraTreesRegressor, word embedding

1. Introduction

RSL17BD (Waseda University Sakai Laboratory) participated in the Third Dialogue Breakdown Detection Challenge (DBDC3) [1] and submitted three runs to both English and Japanese subtasks. Following the approach of Sugiyama [2], we utilise ExtraTreesRegressor [3][1], but instead of his simple word overlap feature, we employ term frequency vectors and word embedding vectors to compute utterance similarities. Given a target system utterance, we use ExtraTreesRegressor to estimate the mean and variance of its breakdown probability distribution, and then derive the breakdown probabilities from them. We took this approach because the use of ExtraTreesRegressor by Sugiyama was successful at the Second Dialogue Breakdown Detection Challenge (DBDC2) [4]. However, they did not utilise term frequency and word embedding vectors as we do.

To calculate word embedding vector similarities between two neighbouring utterances, Run 1 follows the approach of Omari et al. [5] and uses the maximum cosine similarity and geometric mean; Run 3 uses arithmetic mean instead; Run 2 utilises the cosine similarities of all term pairs from the two utterances. Run 2 statistically significantly outperforms the other two for the English data ($p = 0.011$ with Jensen-Shannon Divergence and $p = 0.009$ with Mean Squared Error).

[1] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

2. Prior Art

2.1. Dialogue Breakdown Detection Challenge 2

At DBDC2, systems that analysed different breakdown patterns [2, 6] tended to exhibit high performances [4]. In particular, the top-performing system of Sugiyama [2] employed the following features based on the breakdown pattern analysis, along with a few others: *turn-index*, which denotes where the utterance appears in a dialogue, *utterance lengths* (in characters and in terms), and *term overlap* between the target system utterance and the previous user utterance as well as that between the target system utterance and the previous system utterance. Following Sugiyama, our approach also utilises ExtraTreesRegressor for estimating the breakdown probability of each system utterance.

2.2. Utterance Similarity

Instead of the simple term ovelap feature of Sugiyama [2], we use utterance vector similarities as features for ExtraTreesRegressor. Given two utterances, we compute a cosine similarity based on term frequency vectors [5, 7] and those based on word embedding vectors [5].

3. Proposed Methods

Our methods are comprised of three steps: preprocessing the English or Japanese data, feature extraction, and training and estimation by ExtraTreesRegressor. Below, we describe each step.

3.1. Preprocessing

3.1.1. English data

For English data, we apply Punkt Sentence Tokenizer[2] to break the utterances into sentences, Penn Treebank Tokenizer[3] to tokenise the sentences, and Krovetz Stemmer to stem the tokens. We use a stopword list from the Stopwords Corpus in the nltk dataset[4] when computing term frequency vectors (Section 3.2.2). We also utilise a publicly available pre-trained word embedding matrix[5] for computing word embedding vectors (Section 3.2.3).

[2] http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.punkt

[3] http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.treebank

[4] http://www.nltk.org/nltk_data/

[5] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

3.1.2. Japanese data

For Japanese data, we tokenise utterances and extract the base forms using MeCab[6], and use a stopword list from SlothLib[7]. For training a word embedding matrix, we use Japanese Wikipedia[8].

3.2. Features

We extract features of the target utterances in Table 1 to estimate breakdown probability distributions. The last three features are explained below.

Table 1: *Features*

Feature
turn-index of the target utterance
length of the target utterance (number of characters)
length of the target utterance (number of terms)
keyword flags of the target utterance
term frequency vector similarities among the target system utterance, the immediately preceding user utterance, and the system utterance that immediately precedes that user utterance
word embedding vector similarities among the target system utterance, the immediately preceding user utterance, and the system utterance that immediately precedes that user utterance

3.2.1. Keyword Flag

At DBDC2, we classified system utterances based on predefined cue words such as the question mark [6]. This time, we tried to select such keywords automatically, by using the Robertson/Sparck Jones *offer weight* [8]. Given U, the set of all utterances, and $V(\subseteq U)$, the set of utterances that may be associated with breakdowns (see below for details), we compute the offer weight for term t from V as follows:

$$ow(t,V) = r(t) * \log\left(\frac{(r(t)+0.5)(N-n(t)-R+r(t)+0.5)}{(n(t)-r(t)+0.5)(R-r(t)+0.5)}\right), \quad (1)$$

where N denotes the number of utterances in U, R denotes the number of utterances in V, $n(t)$ denotes the number of utterances in U containing t, and $r(t)$ denotes the number of utterances in V containing t. The terms from V are then sorted by the offer weight, and the top 10 terms are selected as the keywords representing V.

Let A be the number of annotators and let $f(l|u)(\leq A)$ be the number of annotators that assigned label $l \in \{\mathrm{NB}, \mathrm{PB}, \mathrm{B}\}$ to utterance u in the development data. Here, NB means "Not a Breakdown," PB means "Possible Breakdown," and B means "Breakdown." The breakdown probability for u is hence given by $p(l|u) = f(l|u)/A$. The V for Eq. 1 is given by:

$$V = \{u | p(\mathrm{B}|u) \geq p(\mathrm{PB}|u), p(\mathrm{B}|u) \geq p(\mathrm{NB}|u), u \in U\}. \quad (2)$$

That is, V is the set of training utterances for which the majority of the annotators assigned the B label.

In addition to the above V, we also used the following two sets of utterances for extracting 10 keywords based on the offer weight: V', the set of *user* utterances u' *immediately preceding* the system utterance in V; V'', the set of *system* utterances u'' *immediately preceding* the user utterance in V'. That is, system utterance u'' is immediately followed by user utterance u', which in turn is immediately followed by system utterance u. Thus, a total of 30 keywords are extracted from the development data from V, V', and V'' in Table 2-4 for English and in Table 5-7 for Japanese. Given a target system utterance, the presence/absence of these keywords are used as features: we refer to these features as *keyword flags*.

Table 2: *English keywords extracted from the development data from V*

term t	$ow(t,V)$
.	1439.944526
i	233.276525
,	208.164604
n't	180.762577
's	153.848512
'm	120.785285
and	106.350508
the	95.871451
know	91.472349
to	86.660370

Table 3: *English keywords extracted from the development data from V'*

term t	$ow(t,V')$
?	393.339734
are	105.219367
ok	74.840959
what	71.531507
who	45.774402
you	38.739741
name	23.375171
so	21.294739
bot	20.884654
accident	20.856952

3.2.2. Utterance Similarity based on Term Frequency Vectors

Following the approach of Allan et al. [7], given a system utterance u and its immediately preceding user utterance u', we compute the similarity between them based on term frequency vectors as follows:

$$tsim(u'|u) = \sum_{t \in T(u)} TF(t,u)\, TF(t,u') \log \frac{N+1}{n(t)+0.5}, \quad (3)$$

where $T(u)$ denotes the set of terms that occur in u (excluding stopwords), $TF(t,u) = \log(tf(t,u)+1)$ and $tf(t,u)$ is the

[6] http://taku910.github.io/mecab/
[7] http://www.dl.kuis.kyoto-u.ac.jp/slothlib/
[8] https://dumps.wikimedia.org/jawiki/

Table 4: *English keywords extracted from the development data from V''*

term t	$ow(t, V'')$
.	1035.760495
i	156.393348
's	126.150777
,	110.902187
n't	95.852656
...	78.882248
to	77.249165
'm	69.896469
something	64.789772
thought	62.715845

Table 5: *Japanese keywords extracted from the development data from V*

term t	$ow(t, V)$
、	866.027026
の	660.821041
です	656.387194
が	621.109417
で	526.545654
に	466.167975
は	455.930222
。	433.755614
ます	395.145984
を	349.132137

Table 6: *Japanese keywords extracted from the development data from V'*

term t	$ow(t, V')$
か	1266.105337
？	1231.568310
。	440.290135
の	327.413417
は	327.124639
何	268.767274
ます	255.333050
ん	243.654132
好き	235.091141
です	226.581905

Table 7: *Japanese keywords extracted from the development data from V''*

term t	$ow(t, V'')$
、	742.077753
の	580.849514
で	474.075817
が	455.866965
。	449.044624
ます	422.740444
を	391.454682
に	357.000090
する	347.599865
は	344.289120

term frequency of t in u.

Similarly, we compute $tsim(u''|u)$ (i.e., the similarity between u and the preceding system utterance), as well as $tsim(u''|u')$ (i.e., the similarity between the preceding system and user utterances given u). We use all three similarities as features for the given u.

3.2.3. Utterance Similarity based on Word Embedding Vectors

We tried three approaches to computing word embedding vectors to generate Runs 1, 2, and 3.

Run 1 follows the approach of Omari et al. [5], which is based on maximum cosine similarities and geometric mean. Given a system utterance u and the immediately preceding user utterance u', the similarity is computed as follows:

$$Cov(u'|u) = \frac{1}{|W(u)|} \sum_{t_1 \in W(u)} \max_{t_2 \in W(u')} \{csim(t_1, t_2)\} , \quad (4)$$

$$wsim1(u, u') = \sqrt{Cov(u'|u) * Cov(u|u')} , \quad (5)$$

where $csim(t_1, t_2)$ denotes the cosine similarity between the word embedding vectors for t_1 and t_2, computed based on the word embedding matrix described in Section 3.1. $W(u)$ denotes the set of terms which occur in utterance u and are valid for this matrix. Similarly, we compute $wsim1(u, u'')$ (i.e., the word embedding vector similarity with the preceding system utterance) and $wsim1(u', u'')$ (i.e., the word embedding vector similarity between the preceding user and system utterances). All three similarities are used as features for the given u.

Run 3 is similar to Run 1, but uses arithmetic mean instead of geometric mean to obtain a symmetric similarity:

$$wsim3(u, u') = \frac{Cov(u'|u) + Cov(u|u')}{2} . \quad (6)$$

Instead of Eq. 4 that relies only on the maximum word embedding vector similarity for each term from an utterance and for each term from a preceding utterance, Run 2 uses the following symmetric simlarity:

$$wsim2(u, u') = \frac{\sum_{t_1 \in W(u)} \sum_{t_2 \in W(u')} csim(t_1, t_2)}{|W(u)||W(u')|} . \quad (7)$$

That is, this is the average over all word embedding vector similarities concerning terms from an utterance and those from a preceding utterance. This is based on the observation that the maximum-based approach of Omari et al. do not utilise the non-maximum word embedding vector similarities at all.

3.3. Training and Estimation

3.3.1. Training

Our final step is to use the aforementioned features for training ExtraTreesRegressor to estimate the mean and the variance of the distribution of labels for a given system utterance in the evaluation data.

In the training phase, we first map the categorical labels $\mathrm{B}, \mathrm{PB}, \mathrm{NB}$ to integers $-1, 0, 1$. Then, for a given utterance u in the development data, the label frequencies $f(l|u)$ ($l \in \{\mathrm{B}, \mathrm{PB}, \mathrm{NB}\}$) over the A annotators ($\sum_l f(l|u) = A$) yield a probability distribution with mean α and variance β^2, given by:

$$\alpha = \frac{1 * f(\mathrm{NB}|u) + 0 * f(\mathrm{PB}|u) + (-1) * f(\mathrm{B}|u)}{A}, \quad (8)$$

$$\beta^2 = \frac{(1-\alpha)^2 * f(\mathrm{NB}|u) + \alpha^2 * f(\mathrm{PB}|u) + (-1-\alpha)^2 * f(\mathrm{B}|u)}{A}. \quad (9)$$

Next, we train ExtraTreesRegressor with the aforementioned features from the development data and the above means and variances as the target variables.

3.3.2. Testing

Given a system utterance from the evaluation data, its features are extracted as described in Section 3.2, and the trained ExtraTreesRegressor yields the estimated mean $\hat{\alpha}$ and the estimated variance $\hat{\beta}^2$ for the unknown labels for this test utterance. By substituting $p(\mathrm{B}|u) = f(\mathrm{B}|u)/A, p(\mathrm{PB}|u) = f(\mathrm{PB}|u)/A, p(\mathrm{NB}|u) = f(\mathrm{NB}|u)/A$ to Eqs. 8 and 9, we can convert the estimated mean and variance for the test utterance to its estimated label probabilities as follows:

$$\hat{p}(\mathrm{NB}|u) = \frac{\hat{\alpha}^2 + \hat{\alpha} + \hat{\beta}^2}{2} \quad (10)$$

$$\hat{p}(\mathrm{PB}|u) = 1 - \hat{\alpha}^2 - \hat{\beta}^2 \quad (11)$$

$$\hat{p}(\mathrm{B}|u) = \frac{\hat{\alpha}^2 - \hat{\alpha} + \hat{\beta}^2}{2} \quad (12)$$

4. Results

Tables 8 and 9 show the official results of our English and Japanese runs, respectively. In these tables, F1(B) denotes the F1-measure where only the B labels are considered correct (the larger the better); JSD(NB,PB,B) denotes the mean Jensen-Shannon Divergence, and MSE(NB,PB,B) denotes the mean squared error (the smaller the better) [1]. It can be observed that Run 2 seems to have done well on average.

Tables 10-13 show the results of comparing the means (JSD and MSE) of Runs 1-3 based on Tukey's Honestly Significant Differences (HSD) test. The p-values are shown alongside with effect sizes (standardised mean differences) [9]. Tables 5 and 6 show that Run 2 statistically significantly outperforms Runs 1 and 3 in terms of both JSD and MSE for the English data, while Tables 7 and 8 show that none of the differences are statistically significant for the Japanese data. The English results suggest that our approach of retaining the similarity information for all term pairs (Eq. 7) deserves further investigations.

5. Conclusions

We submitted three runs to both English and Japanese subtasks of DBDC. Run 1 used the maximum cosine similarity and geometric mean; Run 3 used arithmetic mean instead; Run 2 utilised the cosine similarities of all term pairs from two neighbouring utterances. Run 2 statistically significantly outperformed the other two for the English data ($p = 0.011$ with Jensen-Shannon Divergence and $p = 0.009$ with Mean Squared Error). However, for Japanese, Run 2 did not statistically significantly outperform the other two.

Our future work includes a comparison of our English and Japanese results to investigate what caused Run 2 to be successful for the English data but not for the Japanese data.

Table 8: *Official results (over English system utterances)*

Run	F1(B)	JSD(NB,PB,B)	MSE(NB,PB,B)
Run 1	0.3126	0.0432	0.0254
Run 2	**0.3201**	**0.0412**	**0.0241**
Run 3	0.3025	0.0426	0.0250

Table 9: *Official results (over Japanese system utterances)*

Run	F1(B)	JSD(NB,PB,B)	MSE(NB,PB,B)
Run 1	0.2635	0.1539	0.0882
Run 2	0.2795	**0.1528**	**0.0879**
Run 3	**0.2844**	0.1543	0.0886

Table 10: *P-values based on the Tukey HSD test/effect sizes for JSD(NB,PB,B) (English)*

	Run 2	Run 3
Run 1	$p = 0.011(0.091)$	$p = 0.636(0.029)$
Run 2	-	$p = 0.116(0.063)$

Table 11: *P-values based on the Tukey HSD test/effect sizes for MSE(NB,PB,B) (English)*

	Run 2	Run 3
Run 1	$p = 0.009(0.093)$	$p = 0.678(0.027)$
Run 2	-	$p = 0.089(0.067)$

Table 12: *P-values based on the Tukey HSD test/effect sizes for JSD(NB,PB,B) (Japanese)*

	Run 2	Run 3
Run 1	$p = 0.882(0.017)$	$p = 0.979(0.007)$
Run 2	-	$p = 0.778(0.024)$

Table 13: *P-values based on the Tukey HSD test/effect sizes for MSE(NB,PB,B) (Japanese)*

	Run 2	Run 3
Run 1	$p = 0.961(0.010)$	$p = 0.956(0.010)$
Run 2	-	$p = 0.845(0.020)$

6. References

[1] R. Higashinaka, K. Funakoshi, M. Inaba, Y. Tsunomori, T. Takahashi, and N. Kaji, "Overview of dialogue breakdown detection challenge 3," in *Proceedings of Dialog System Technology Challenge 6 (DSTC6) Workshop*, 2017.

[2] H. Sugiyama, "Chat-oriented dialogue breakdown detection based on the analysis of error patterns in utterance generation (in Japanese)," in *SIG-SLUD-B505-22*. The Japanese Society for Artificial Intelligence, Special Interest Group on Spoken Language Understanding and Dialogue Processing, 2016, pp. 81–84.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] R. Higashinaka, K. Funakoshi, M. Inaba, Y. Arase, and Y. Tsunomori, "The dialogue breakdown detection challenge 2 (in Japanese)," in *SIG-SLUD-B505-19*. The Japanese Society for Artificial Intelligence, Special Interest Group on Spoken Language Understanding and Dialogue Processing, 2016, pp. 64–69.

[5] A. Omari, D. Carmel, O. Rokhlenko, and I. Szpektor, "Novelty based ranking of human answers for community questions," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016, pp. 215–224.

[6] S. Kato and T. Sakai, "Dialogue breakdown detection based on word2vec utterance vector similarities (in Japanese)," in *SIG-SLUD-B505-20*. The Japanese Society for Artificial Intelligence, Special Interest Group on Spoken Language Understanding and Dialogue Processing, 2016, pp. 70–71.

[7] J. Allan, C. Wade, and A. Bolivar, "Retrieval and novelty detection at the sentence level," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003, pp. 314–321.

[8] S. Robertson and K. Spärck Jones, "Simple, proven approaches to text retrieval," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-356, Dec. 1994.

[9] T. Sakai, "Statistical reform in information retrieval?" *SIGIR Forum*, vol. 48, no. 1, pp. 3–12, 2014.