オンライン掲示板の反応予測に有効なテキスト素性の分析

狩野 竜示　三浦 康秀　大熊 智子
富士ゼロックス株式会社
{kano.ryuji, yasuhide.miura, ohkuma.tomoko}@fujixerox.co.jp

1 はじめに

オンライン掲示板をはじめとするソーシャルメディアにおけるユーザーの反応度を予測する研究は，話題となるニュースの予測，効果的な宣伝方法の解明等に応用可能である事から，注目を集めている．中でも，Reddit[1] [2]は，規模が大きく，データが公開されている事から，研究対象となる事が多い [1, 2]．Reddit では多様な話題に関するテキストが投稿され，ユーザーはその投稿に対し，支持か不支持かの評価を行う事ができる．支持の票数から不支持の票数を引いたものは KarmaScore と呼ばれる．テキストやその他の属性から，KarmaScore を予測した先行研究は数多く存在する [1, 2]．しかし，KarmaScore に影響するテキスト素性を分析した論文はこれまでに知られていない．本稿では予測対象のテキストのみならず，返信等の周辺のテキストを素性として加え，Attention 機構 [3] を利用する事で，複数のテキスト素性の内，KarmaScore の予測に有効な素性の分析を行った．

2 関連研究

ソーシャルメディアにおけるユーザー反応度を予測した研究は数多く存在し，動画サイト [4] や，マイクロブログ [5, 6] を対象にしたものが知られている．

テキストやその他の属性から Reddit の KarmaScore を予測する先行研究には，投稿群の構造を複数の潜在変数ベクトルの線形和で表現するモデルを使用したもの [1] や，話題を複数潜在変数ベクトルの線形和として表現し，LSTM に逐次的に入力する言語モデルを使用したもの [2] がある．

Attention 機構は元来翻訳などの文生成タスクに使用されてきたが，これを分類モデルに適用すると，分類に寄与する素性の抽出が可能となる [3]．本稿ではこれを踏襲し，KarmaScore の予測に寄与するテキスト素性を分析可能な，Attention 機構を持ったモデルを構築した．

[1]Reddit，http://www.reddit.com/

[2]Reddit および、Reddit ロゴは、reddit inc.(レディット・インコーポレイテッド) の登録商標 (第 5472080 号) です。

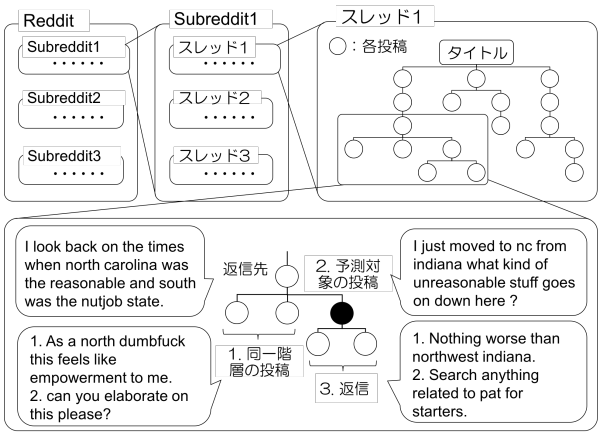

図 1: Reddit のスレッドの模式図

3 データ

3.1 Reddit の構造

図 1 に Reddit の構造を示す．Reddit には，投稿をトピック毎にまとめた Subreddit が存在する．各 Subreddit には，スレッドと呼ばれる投稿群の単位が存在する．ユーザーが新しい話題に関する投稿を最初に行う．これをタイトルと呼ぶ．タイトルには返信が可能であり，その返信にも返信する事が可能である．タイトルと，その下部にある全ての投稿を 1 つのスレッドと呼ぶ．各投稿には複数の返信が可能であるため，スレッドの構造は，図 1 右上部のような木構造になる．この時，円は各投稿を表しており，線で結ばれた 2 つの円は，上部のものが返信先，下部のものが返信の関係にある．

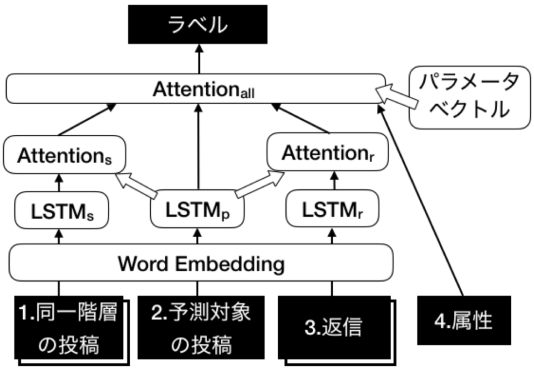

図 2: モデル図

3.2 分析用データ

本研究では，[2] のデータを分析対象とした[3]．使用されている Subreddit は，AskWomen，AskMen，politics の 3 つであり，各 Subreddit における投稿数は約 72 万，約 97 万，約 195 万である．これらを 3:1:1 に分割し，それぞれ，学習データ，開発データ，評価データとした．

4 手法

4.1 素性

図 1 下部に，各テキスト素性の関係性を模式的に示す．素性は以下の 3 つである．1. 予測対象の投稿と同一階層にある投稿，2. 予測対象の投稿，3. 予測対象の投稿についた返信，である．「2. 予測対象の投稿」は 1 つであるが，「1. 同一階層の投稿」 と「3. 返信」は 0 以上の整数個存在しうる．

テキスト素性の他に，スレッド中の深さ，前回の投稿時間からの相対時間などを含む 13 の属性を素性として利用した．13 の内訳は，[2] と同様である．

4.2 モデル

図 2 にモデルの概要を示す．黒の矩形はデータを，白の矩形は関数を，黒い矢印はデータの入出力を示す．また，白の矢印は，Attention の重み計算に用いた事を示している（後述）．4.1 節で述べた 1. から 3. のテキストを単語毎に分割し，ID を付与する．これらの配列を Word Embedding 層に投入し，単語ベクトルの配列を得る．ここで，Word Embedding 層は 3 種類の投稿で共通である．単語ベクトルの配列をそれぞれ，$X_{si} = x_{si1}... \ x_{siT}, X_p = x_{p1}... \ x_{pT}, X_{rj} = x_{rj1}... \ x_{rjT}$ とする．$i = 1..N_s$，および $j = 1..N_r$ はそれぞれ複数ある同一階層の投稿，および返信の番号を，T は系列長を示している．これらの単語ベクトルを LSTM にて系列処理を行う．LSTM に単語ベクトル x_t を入力した時，隠れ層の状態 h_t は以下のように計算される．

$$
\begin{aligned}
h_t &= \tanh(c_t) \odot o_t \\
c_t &= c_{t-1} \odot f_t + g_t \odot i_t \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
g_t &= \tanh(W_g x_t + U_g h_{t-1} + b_g)
\end{aligned}
$$

c_t は記憶素子，i_t は入力ゲート，f_t は忘却ゲート，o_t は出力ゲート，g_t は状態候補，$\odot$ は要素積を示している．W，U，b はパラメータである．

LSTM は 1. から 3. のテキストごとに異なるモデルを用いている．これを，$\mathrm{LSTM_s}, \mathrm{LSTM_p}, \mathrm{LSTM_r}$ とする．各単語ベクトルの配列 X_{si}, X_p, X_{rj} を各 LSTM に入力した後の隠れ層の状態 h_{siT}, h_{pT}, h_{rjT} をそれぞれ各テキストの中間表現として用いる．以後，添字の T は省略する．4.1 節の通り，「1. 同一階層の投稿」と「3. 返信」は複数存在しうる．そのため，これらについては Attention 機構を使用して，複数ある投稿の内，どの投稿の素性が最も有用かを選択するように学習した．該当する投稿が存在しない時，それを示すトークンで代替した．この時，Attention の重みの計算には，「2. 予測対象の投稿」の中間表現 h_p を用いている．「1. 同一階層の投稿」の中間表現の配列 $h_{s1}... \ h_{sN_s}$ を $\mathrm{Attention_s}$ に入力した時に得られるベクトル s は以下のように計算される．

$$
\begin{aligned}
s &= \sum_{k=1}^{N_s} \alpha_k u_k \\
\alpha_k &= \frac{exp(h_p^T u_k)}{\sum_{i=1}^{N_s} exp(h_p^T u_i)} \\
u_i &= \tanh(W_s h_{si} + b_s)
\end{aligned}
$$

同様に，「3. 返信」の中間表現の配列 $h_{r1}... \ h_{rN_r}$ を $\mathrm{Attention_r}$ で計算し，ベクトル r を得る．これによって，1.，2.，3. それぞれに対応する素性， s, h_p, r を得た．3 つのテキスト素性と，属性素性を $\mathrm{Attention_{all}}$ によって，重み付け線形加算し，全結合層によって次元圧縮する事で，最終的にラベルを出力している．

[3] https://github.com/hao-cheng/factored_neural/

$\text{Attention}_{\text{all}}$ の計算式は，上式 $\text{Attention}_{\text{s}}$ のものと同一であるが，重みの計算に必要な h_p に対応するベクトルは，[3] 同様，パラメータであるベクトルを使用した．

探索したハイパーパラメータは，3 つの Subreddit で同一であり，候補が複数ある場合は GridSearch を行い，最も精度の高いものを採用した．Word Embedding の次元は 50 あるいは 100，LSTM の隠れ層の次元は 64 あるいは 128，ミニバッチサイズは 64，Dropout の比率は 0.5，epoch 数は 5，投稿の単語長さが 50 以上となる場合は，最初の 50 単語のみを使用した．頻度順に上位 32000 語に単語 ID を振り分け，それ以外の単語と URL は，それぞれトークン <unk> と <url> に置換した．最適化は Adam で行い，ハイパーパラメータは，$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ とした．評価は 1000iteration 毎に行い，開発データでの精度が最も良い時の評価データの精度を結果として用いた．実装は全て Chainer[4]で行った．

5 実験

5.1 タスク設定

Reddit の KarmaScore の分布は Zipf の法則に従う [1]．このような偏った分布に従う KarmaScore をクラス分類する手法が提案されている [1, 2]．本稿では，これに則り，KarmaScore を 8 つのクラスに離散化した．まず，KarmaScore が 1 以下のものを $Class_1$ とする．$Class_k (k = 1...i)$ の和集合の補集合を $ClassR_i$ とする．$ClassR_i$ に含まれるデータの内，中央値 m_i 未満のものを $Class_{i+1}$ とし，m_i 以上のデータを $ClassR_{i+1}$ とした．これを i=1 から i=6 まで繰り返し，$Class_1$ から $Class_7$ までを定義した．そして，$ClassR_7$ を $Class_8$ と定義する事で KarmaScore を 8 つのクラスに離散化した．以下に，模擬アルゴリズムを示す．

Algorithm 1 KarmaScore の離散化

$Class_1 = \{x_j \in AllData | KarmaScore(x_j) \leq 1\}$
for $i = 1$ to $N - 2$ **do**
  $ClassR_i = \overline{\bigcup_{k=1}^{i} Class_k}$
  $m_i = Median(ClassR_i)$
  $Class_{i+1} = \{x_j \in ClassR_i | KarmaScore(x_j) \leq m_i\}$
end for
$Class_N = \overline{\bigcup_{k=1}^{N-1} Class_k}$

[4]Chainer，https://chainer.org/

本稿では，このように離散化された KarmaScore の分類タスクを行った．評価の際は，$\bigcup_{i=1}^{k-1} Class_i$ と $\bigcup_{j=k}^{8} Class_j$ を二値分類するタスクにおいて，F1Score を，k=2 から k=8 まで計算し，その平均値を評価値とした．

5.2 結果

各 Subreddit における分類精度を表 1 に示す．記載の値は 5.1 節で述べた，F1Score の平均値である．FNN は [2] で提案された State-of-the-art の手法である．BoW は BagOfWords を，属性のみは，先述した 13 の属性データのみを入力としたモデルを指し，それぞれ全結合ニューラルネットによってラベルを出力している．全てのモデルで属性素性は使われており，テキスト素性としては，BoW では「2. 予測対象の投稿」のみが，FNN ではそれに加えて「3. 返信」の素性が使われている．Att all は本稿における図 2 のモデル，Concat all は図 2 のモデルの $\text{Attention}_{\text{all}}$ を全結合層に置き換えたものである事から，Attention 機構が精度向上に寄与している事がわかる．

表 1: 各モデルにおける平均 F1Score

	AskMen	AskWomen	politics
Att all	0.534	0.556	0.539
Concat all	0.525	0.552	0.536
FNN[2]	0.527	0.563	0.548
BoW[2]	0.509	0.531	0.518
属性のみ [2]	0.493	0.536	0.513

6 考察

6.1 結果に有効な素性

Attention 機構の各素性の重み係数を分析する事で，どの素性の寄与度が高いかを明らかにする事ができ

表 2: 各 Subreddit の各素性の平均重み

	同一階層	返信	予測対象の投稿	属性
AskMen	10.6%	11.1%	13.9%	64.4%
AskWomen	11.3%	10.6%	11.3%	66.9%
politics	11.9%	11.0%	9.8%	67.4%

表 3: 各 Subreddit の周辺の投稿数

	同一階層の平均投稿数	平均返信数
AskMen	1.16	1.51
AskWomen	1.18	1.59
politics	1.24	2.02

る．本稿では，図 2 の $\text{Attention}_{\text{all}}$ 層において，重みの数値を分析した．Subreddit 毎に，各素性 1. から 3. にかけられた重みの平均値を表 2 に示す．Reddit の KarmaScore 予測においては，属性の寄与が非常に大きい事が知られている [1, 2]．本稿の実験においても，属性の重みが最も高い．テキスト素性を比較すると，AskMen と AskWomen においては，「2. 予測対象の投稿」の重みが高くなっているが，politics においては「1. 同一階層の投稿」や「3. 返信」と比べ，相対的に「2. 予測対象の投稿」の重みは低くなっている．これは，議論が活発化しやすい politics においては，予測対象のテキストよりも周辺のテキストが予測に寄与する事を示唆している．表 3 に，本稿で用いたデータの内，各投稿の同一階層の投稿数，返信数の平均値を示す．これらの値が高いほど，予測モデルにおけるそれぞれの素性の重みも高くなっている．

表 4 に，返信の重みが最も高くなった時のテキスト例を各 Subreddit 毎に示す．重みが強い時，「3. 返信」は極性の高い文で構成されている．この傾向は「1. 同一階層の投稿」についても同様であった．「2. 予測対象の投稿」の重みが最も強くなった時は，投稿が短文であるケースが上位を占めた．

7 おわりに

本研究では，Reddit における KarmaScore の予測を階層性 Attention 機構を用いて行い，予測への寄与度が高いテキスト素性の分析を行った．Subreddit の内，議論が活発になる politics では，同一階層の投稿と，返信が予測により寄与し，他の 2 つの Subreddit では，予測対象の投稿がより寄与する事が判明した．また，Attention 機構を分析する事で，KarmaScore の予測に寄与する返信は極性が強いものである事が判明した．今後は，モデルの精度改善を進め，要約や重要文抽出に適用していく．

参考文献

[1] Hao Fang, Hao Cheng, Mari Ostendorf. Learning Latent Local Conversation Modes for Predicting Community Endorsement in Online Discussions. ACL 2016.

[2] Hao Cheng, Hao Fang, Mari Ostendorf. A Factored Neural Network Model for Characterizing Online Discussions in Vector Space. EMNLP 2017.

[3] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy. Hierarchical Attention Networks for Document Classification. NAACL HLT 2016.

[4] Haitao Li, Xiaoqiang Ma, Feng Wang, Jiangchuan Liu and Ke Xu, On popularity prediction of videos shared in online social networks. CIKM13 2013.

[5] Liangjie Hong, Ovidiu Dan, Brian D. Davison. Predicting Popular Messages in Twitter. In Proc. WWW 2011.

[6] Ruocheng Guo,Elham Shaabani ,Abhinav Bhatnagar, Paulo Shakarian. Toward order-of- magnitude cascade prediction. In Proc ASONAM 2015.

表 4: 返信に最も重みが強い投稿例（太字は極性の強いフレーズ）

Subreddit	投稿	返信	重み係数
AskWomen	Don’t believe that you have an infinite amount of time to do all the things you want to do.	This is so **sad**.	41.8%
AskMen	<url> He loved this inhabiting post. He is proudly humble as dsm as it is having found crossfit	Oh **wow**! Thats him! How did you find this and put the connection together? Holy fucking.	42.4%
politics	Theres a lot of republicans here.	Reminds me of Scott Walkers last election victory. **I love it**. Mods havent stopped crying	38.4%