A Convolutional Neural Network for Ranking Advice Quality in Texts for a Motivational Dialogue System

Patrycja Swieczkowska, Rafal Rzepka and Kenji Araki
Graduate School of Information Science and Technology, Hokkaido University, Japan
{swieczkowska,rzepka,araki}@ist.hokudai.ac.jp

Abstract

In this paper, we present research into advisory texts which eventually will be used to create a dialogue system providing motivational support to the user. We studied advisory comments from an online platform Reddit, including those containing motivational advice. Utilizing advice features identified in previous studies, we were able to correctly rank these comments within groups of three based on the quality of their advice content. Our convolutional neural network achieved mean accuracy of 0.97 in 10-fold cross validation experiments. The contributions of this research are gaining further insight into advice features possessed by advisory comments and creating a novel way of ranking advisory texts.

1 Introduction

Lack of motivation is an important issue and there have been numerous studies on the topic conducted in professional [Badubi, 2017; Gerhart and Fang, 2015] and academic [Elmelid *et al.*, 2015, Litalien *et al.*, 2015] settings, as well as within the context of mental health issues [Fussner *et al.*, 2018; Hershenberg, 2017]. However, up till today there were few experiments involving motivational dialogue agents. Therefore, the main goal of our research is to create a dialogue system that would be able to motivate the user to perform their everyday tasks. It was already established that creating such a system is not trivial [Swieczkowska *et al.*, 2017]. Previous studies [Swieczkowska *et al.*, 2018] have identified 14 features that distinguish advisory texts from regular ones. It was successfully proven that there are significant differences in feature scores between these two types of texts and that these features can be used to classify online user comments as motivational/advisory or regular. This was done to create a classification and selection algorithm for data that will then be used in training and testing a motivational dialogue system.

In this paper, we describe further research involving the 14 advice features. They were chosen through a quality analysis of online comments containing advice; details are given in section 2.2. of this paper. Since they proved to be useful in selecting texts with advisory content, we now studied their correlation with quality of the advice contained in the text. Using the same data as previous studies, namely online comments provided by users of Reddit[1], we created a neural network that ranks the quality of advice texts within groups of three. In other words, given any 3 advice texts, our algorithm is able to select the best one according to points awarded by Reddit users. That advice can then be given to the user of the end-goal motivational dialogue system. For example, in response to the user's problem, the system can produce 3 different candidate advice texts (by choosing appropriate advice sentences from a corpus) and then use the ranking component to select the best one. The ranking algorithm is one of the main contributions of this paper; another one is deeper insight into the 14 advice features and their relation to advice quality.

The paper is organized as follows. Section 2 describes related work in the field of motivation in dialogue systems as well as text ranking. Section 3 presents our datasets and features. Section 4 describes the architecture of our system. Section 5 presents the details of our experiments and their results. Section 6 provides error analysis and discussion about our findings. Section 7 concludes the paper.

2 Related Work

2.1 User Motivation

There are numerous studies suggesting approaches to influencing motivational states in users; however, few of them contain actual experiments. Most of them propose frameworks without verifying their usefulness (for example [Callejas and Griol, 2016] or [He *et al.*, 2010]). Papers describing empirical studies include research on motivating users to do indoor cycling every day for a specified period of time with a robot companion [Süssenbach *et al.*, 2014] or encouraging users to perform longer planking exercise by giving them acknowledging feedback from a robot that exercised together with them [Schneider and Kummert, 2016]. However, authors of these studies scripted the agent dialogue and limited it to a handful of topics relevant to the task. Since both studies dealt only with exercise, their very specific findings

[1] https://www.reddit.com/

cannot be generalized to other everyday activities. In contrast, we plan our system to not be limited to one or a few topics. Among studies not involving exercise, Kaptein *et al.* [2012] describe a study where subjects were being persuaded to reduce snacking via personalized short text messages. The messages were tailored to the user based on the user's score on the Susceptibility to Persuasion Scale [Kaptein *et al.*, 2009]. However, these messages were again crafted by the researchers and involved no natural language processing. Our goal is to create a system that produces motivational advice by itself, composing it from fragments of highly rated advisory texts obtained with crowdsourcing.

2.2 Text Ranking

Studies concerning text ranking usually involve criteria like relevance to the user's query and are conducted as part of research in information retrieval. Documents in a given database are ranked according to their usefulness in providing the user with information about a particular topic. Most recent developments in this field include improving the tf-idf weighted ranking method with association rules [Jabri *et al.*, 2018], incorporating user browsing patterns into sorting query search results [Sethi and Dixit, 2018] and utilizing Hadoop and MapReduce platforms to improve search precision [Malhotra *et al.*, 2018]. However, these studies are only partially relevant to our research problem. An effective document ranking algorithm, such as the ones mentioned above, would be useful in retrieving appropriate advice for the user based on their query and will be a point of focus for our next step. In contrast, this paper describes an algorithm for ranking advice quality, which is a different issue and therefore a different approach must be used. The main difference is that our method ranks the texts against each other rather than by relevance to some external search term.

There are also numerous papers describing approaches to ranking texts for purposes other than answering the user's query. Fang *et al.* [2017] present a sentence ranking algorithm for extractive text summarization. Vajjala and Meurers [2016] rank sentences in the text based on their readability while studying text simplification. Myangah and Rezai [2016] rank Persian texts based on their vocabulary richness and use this information to determine the genre of the text. However, to the best of our knowledge, there is no proposed algorithm for ranking advice texts based on advice quality.

3 Datasets and Features

2.1 Datasets

As our source of advice texts, we have used the online discussion platform Reddit. It is a place where people share opinions, discuss matters or ask for advice on different topics. The platform is divided into numerous so-called subreddits, each with a different purpose and topic. A user can post in any appropriate subreddit and other users can comment on their post. Users can also vote on both posts and comments.

Subreddit	Posts #	Comments #
r/getdisciplined	624	1,872
r/Advice	3,066	5,850
Total	3,690	11,070

Table 1: Datasets used in our study

In our experiments, we have studied comments downloaded specifically from subreddits where authors of posts ask for advice, so the comments are bound to contain advisory content. This data was chosen because such user posts are closest to what we imagine as input to our system while other users' comments are closest to the ideal output. The subreddits we used were r/getdisciplined[2], where people ask for motivational advice, and r/Advice[3], where people ask for general advice on a variety of topics. From each subreddit, we obtained posts, each with 3 best rated comments; such a comment triple was our basic unit of data. This method ensured that all comments within the triple contained advice on the same topic and as such could be compared against each other. Table 1 present the breakdown of the amount of data downloaded from each subreddit.

2.2 Features

We have utilized the 14 advice features that proved useful in previous studies [Swieczkowska *et al.*, 2018]. The features were determined through a quality analysis of top r/getdisciplined comments, which were the closest to ideal data used in previous research. Such comments usually contained imperative or advice expressions and their content was rather specific. The authors of the comments also said that they related to the problem personally. We coded these qualities into Imperative Score, Advice Score, Specificity Scores and Relatability Score described below. We then added sentiment analysis using Sentic scores to complete the list of features.

All comments were pre-processed by detecting sentence boundaries and assigning part-of-speech tags. For some feature calculations, we also removed stopwords. In the following paragraphs, *wordlist_withstops* means a list of all words in the comment, *wordlist_nostops* means the same list with stopwords removed and *sent_list* means a list of sentences in the comment. The features were calculated on each comment separately in the following manner.

Sentics scores of **aptitude, attention, pleasantness** and **sensitivity** were measured automatically using the Sentic library for Python[4] on *wordlist_nostops*. The library is an API to the SenticNet knowledge base [Cambria *et al.*, 2018] containing information on sentiment values of words. All the sentics values fall on the scale between -1 and 1.

[2] https://www.reddit.com/r/getdisciplined/

[3] https://www.reddit.com/r/Advice/

[4] https://pypi.org/project/senticnet/

Relatability score was measured by the percentage of first-person pronouns, including possessive pronouns, in *word-list_withstops*. The score range was 0 to 1.

Imperative score was measured by the percentage of imperative expressions in the comment text. Specifically, we looked for expressions such as clauses beginning with non-infinitive verbs, the word *please* preceding a verb and phrases comprised of *you* or *OP* (which stands for *Original Poster* and is a popular way of referring to the author of the post on Reddit) and a modal verb. We excluded sentences that started with auxiliary verbs *do* and *have* to avoid counting in questions with syntax similar to imperative expressions. We counted the percentage on number of all words divided by 2, since most of the imperative expressions we looked for were bigrams. The score range was 0 to 1.

Advice Score was defined as the number of advisory expressions in the text. For this purpose, we prepared a list of possible advisory expressions, including phrases like *you need to*, *it might be worth*, or *if I were you*, along with words like *recommend* or *suggest*. The full list is given in Table 2. This feature also counted website links, which we discovered to be a way of offering advice in many comments in our datasets. In the preprocessing stage we replaced all links with the token *[link]*, which then also counted as an advisory expression. The overall comment Advice Score was divided by 10 to scale it down to the level of other features.

Specificity Scores that included six different features. They were first proposed by Deshpande *et al.* [2010] to extract suggestions and complaints from employee surveys and online product reviews. The goal was to find sentences containing specific content. We adapted these features into our study because initial analysis showed that comments containing specific advice were among the best rated on any given post. The calculations were performed for each sentence in the comment using *sent_list*, and the final scores for the entire comment were obtained by adding up all the sentence scores. The features were: **Average Semantic Depth (ASD)**, **Average Semantic Height (ASH)**, **Total Occurrence Count (TOC)**, **Count of Named Entities (CNE)**, **Count of Proper Nouns (CPN)** and **Sentence Length (LEN)**.

We only slightly modified the calculations provided by [Deshpande et al., 2010]. For both ASD and ASH, we had to retrieve hypernymy/hyponymy hierarchies from WordNet ontology[5] for each content word (meaning nouns, verbs, adjectives and adverbs). For each word, the longest path in the hierarchy that led from the word to its highest hypernym determined the Semantic Depth of that word. Similarly, the shortest path from the word to its lowest hyponym determined its Semantic Height. To obtain the ASD score for the entire sentence, we added all the ASDs for all the content words and divided the sum by the total number of content words in the sentence (by which we mean a sentence from *sent_list* with stopwords removed). We performed the same calculations for ASHs.

you need to	*have you thought about*
op needs to	*have you tried*
you have to	*how about*
op has to	*if I were you*
it might be worth	*recommend*
I would	*suggest*
it would be good to	*advise*
it might be good to	*you could always*
you had better	*have you considered*
you'd better	*why not*
your only option is	*why don't you*
[link]	

Table 2: A complete list of advice expressions used by the Advice Score feature.

In contrast to [Deshpande *et al.*, 2010], we added a new feature **ASHD**, which combined Average Semantic Depth and Average Semantic Height by deducting the overall ASH score from the overall ASD score for each comment. This was done to reflect the difference between the two features, which were found to be important in previous studies.

Another change compared to [Deshpande *et al.*, 2010] was that we did not change all content words into nouns before looking them up in WordNet. Nominalization was supposed to help with the lookup, as in 2010 the WordNet ontology was rather developed for nouns but scarce for any other parts of speech. We felt no need to do this anymore in 2019 because WordNet grew immensely since that time. We only lemmatized the words.

Total Ocurrence Count (TOC) meant the number of times the word occurs in the WordNet ontology. We measured it by obtaining occurrence count from WordNet for each lemmatized content word and adding up three lowest scores in a sentence. Count of Named Entities (CNE) meant the number of named entities in the sentence, determined with an NLTK Named Entities tagger[6]. Count of Proper Nouns (CPN) was measured by the number of proper nouns (tagged as NNP, NNPS or CD) in the sentence with stopwords removed. CD stands for Cardinal Number and as such is not a proper noun, but it was included in calculations provided by [Deshpande *et al.*, 2010], so we decided to keep it. Sentence Length (LEN), was the number of words in the sentence with stopwords removed.

We divided ASD, ASH, ASHD, TOC and LEN by 100 and CNE and CPN by 10 to put the scores in the same numerical range as other features.

Table 3 contains an example taken from the r/getdisciplined portion of our dataset. Both the original post and its three comments are included, as well as their respective feature scores and general scores given to them by Reddit users. To conserve space in the table, we removed the new line breaks, but otherwise we kept the text intact.

In addition to our 14 advice features, we used word2vec word embeddings of the texts. Specifically, we obtained a vector

[5] https://wordnet.princeton.edu

[6] https://www.nltk.org/

Post text				Post score: 15		
I am addicted to sleeping. I think the reason for that is because I cannot tolerate my thoughts and the real world. But after spending years like this, I feel awful for sleeping so much. It's not like I sleep 15 hours a day but this habit of mine leads to being absent for classes twice a month and skipping half of gym sessions. Above all I don't bother to improve my life style. With this attitude of mine seeing any kind of future for myself is impossible! Can you give me tips and suggestions how to overcome this bad addiction? If you introduce a reading source, also I would appreciate it a lot. Edit: I don't sleep 15 hours a Day but I am sure I am addicted to sleeping!						
Comment text				Comment score: 20 (rank 0)		
If you're sleeping 15 hours a Day regularly for no apparent reason you need to see a doctor						
aptitude	attention	pleasantness	sensitivity	Relatability score	Imperative score	Advice score
0.094	-0.119	0.123	-0.035	0.000	0.000	0.100
ASD	ASH	ASHD	TOC	CNE	CPN	LEN
0.157	0.152	0.005	0.000	0.000	0.000	0.090
Comment text				Comment score: 7 (rank 1)		
You associate your sleeping to not tolerating your thoughts. To achieve higher capacity in managing your thoughts, have you heard of Mindfulness work? It's a simple technique with effects showing already after a short while.						
aptitude	attention	pleasantness	sensitivity	Relatability score	Imperative score	Advice score
0.179	0.245	0.136	-0.013	0.000	0.048	0.000
ASD	ASH	ASHD	TOC	CNE	CPN	LEN
0.460	0.364	0.096	0.000	0.000	0.000	0.180
Comment text				Comment score: 3 (rank 2)		
I can relate a little bit, as I too love sleep and try to avoid being alone with my own thoughts. I still love sleep, but finding podcasts I really like has helped me with the avoiding my thoughts part. Then I can use them as a bribe to myself..."I can only listen to this on my drive to work/walk to class." "I can only listen to this one at the gym." YMMV, but if you can find an addictive one, or one you find genuinely funny/entertaining, the bribery works. And then if you are one of those "I'm fine as long as I GET there" people for class/gym/work, you can look forward to the getting there part.						
aptitude	attention	pleasantness	sensitivity	Relatability score	Imperative score	Advice score
-0.041	0.029	-0.084	0.114	0.100	0.042	0.000
ASD	ASH	ASHD	TOC	CNE	CPN	LEN
1.188	1.106	0.082	0.080	0.000	0.400	0.540

Table 3: A single data example from our dataset.

for each word in the text and took the average to represent the entire text. The word2vec embeddings had 100 dimensions. We concatenated them with our advice features, ending up with 114 features in total for each comment text.

4 System Architecture

The basic unit of our data was a triple of comments coming from the same post. Each comment had been rated by users, so by comparing their scores we were able to rank the comments with numbers 0, 1 and 2, where 0 represented the best rated comment and 2 represented the lowest rated one. The ratings were not representative across the entire dataset; for example, a comment ranked 0 in its own comment triple may have been ranked 2 in a different triple (given they pertained to the same topic). However, this was not an issue, since our purpose was to select the best comment in the given fixed set of three.

Each comment text had 114 features. To construct our input data, in each comment triple we concatenated the feature vectors into one vector of length 342 (=3x114). Before concatenation, we shuffled each triple and obtained all 6 permutations of their order. Therefore, each triple was present in the dataset 6 times, each time with different order of comments. This was done to lessen the impact of order on the results of the network. As output, the network produced a vector of length 3, where each position gave a number 0, 1 or 2 depending on the rank and order of the comments. For example, if the first 114 features of the input vector represented a comment of rank 2, the next 114 features represented a comment of rank 0 and the last 114 represented a comment of rank 1, then the expected output was a vector of [2, 0, 1].

To ensure that each comment text went through the same initial calculations, we constructed a convolutional neural network. The first layer had 342 units that matched our input vector. Then, we used a filter of length 114 and stride of 114, which essentially meant that each set of 114 comment fea-

Layer	Units #	Output shape
Input	---	(m, 1, 1, 342)
Conv1	114*(1, 114)	(m, 114, 1, 3)
Reshape	---	(m, 3, 1, 114)
Conv2	3*(1, 3)	(m, 3, 1, 38)
Reshape	---	(m, 3, 38)
Fc1	20	(m, 3, 20)
Fc2	10	(m, 3, 10)
Fc3	3	(m, 3, 3)
Reshape	---	(3m, 3)
Argmax	---	(3m, 1)

Table 4: Overview of the network. *Conv* stands for convolutional layers and *Fc* stands for fully connected layers. For convolutional layers, the number of units is the number of filters multiplied by filter size used on the layer.

tures went through the same filter. This ensured that no matter the order of the comments, each one received equal treatment and had equal chances of being assigned any of the three ranks.
On top of the first convolutional layer, we had a second one followed by three fully connected layers. Table 4 gives an overview of the network along with data shape produced by each layer and operation. As we conducted the research using PyTorch, the order of dimensions for convolutional layers follows the PyTorch convention, which is: depth (= number of channels), height, width. The kernel size gives only height and width; depth is exactly the same as the input to the given layer.
We reshaped the output of the first convolutional layer before passing it on to the next layer. The first layer gave output of shape (114, 1, 3) for each data entry. Essentially, this was a vector of length 3, where each position represented one comment and had a depth of 114, because each comment had been convolved by all 114 kernels. We reshaped the output into (3, 1, 114) and fed it as input to *Conv2*. This way, the kernels in the second convolutional layer operated on a vector of length 114, where each position represented one *Conv1* kernel and had a depth of 3 representing the three comments. Each *Conv2* kernel processed three of the 114 positions (with each position incorporating calculations from all comments), yielding 38 results from the processing. The point of this reshaping operation was to allow the *Conv2* kernels to process subsets of *Conv1* kernel results with all three comments each instead of subsets of comments with all 114 *Conv1* kernel results each. It was important to include information about all three comments for each *Conv2* kernel operation, because our results are dependent on all feature relationships within the comment triple. We then reshaped the *Conv2* layer results to reduce the number of dimensions from four to three so that we could pass them to a fully connected layer.
Each layer had the tanh activation function except for the last one, which used a softmax. The output shape from the last layer was (m, 3, 3) where m represents batch size. Essentially, for each training example there were three comments to rank and each of these comments received its own softmax vector

Fold	Training loss	Training accuracy	Test loss	Test accuracy
1	0.016	0.995	0.037	0.991
2	0.038	0.989	0.056	0.984
3	0.276	0.903	0.423	0.889
4	0.031	0.992	0.049	0.989
5	0.072	0.980	0.081	0.975
6	0.060	0.990	0.059	0.987
7	0.084	0.947	0.166	0.931
8	0.032	0.994	0.043	0.992
9	0.020	0.993	0.052	0.989
10	0.036	0.989	0.066	0.986
Average	0.067	0.977	0.103	0.971

Table 5: Results of loss and accuracy values across all folds.

of length 3, where the rank was indicated by the position of 1. For example, if the softmax produced output of [0, 1, 0] for a comment, that comment got rank 1, and if the output was [0, 0, 1] then the comment got rank 2. Therefore, for each training example the output shape was (3, 3) where the first 3 represents the three comments and the second 3 represents the length of softmax. We then reshaped this output so that each comment became its own entry (shape of (3m, 3)). At the very end, we used the argmax function to reduce the output to shape (3m, 1), meaning one rank for each comment in the dataset.

5 Experiments and Results

5.1 Experiment Setup

We trained the network for 1000 epochs using the Adadelta [Zeiler, 2012] optimization algorithm with no changes to its default hyperparameters (this means that we did not set a learning rate manually). We also divided our training data into minibatches of 512. These hyperparameters were decided based on performance.
Overall, we had 22,140 examples in our dataset (total of 3,690 downloaded triples where each triple was present in the dataset 6 times). We performed 10-fold cross validation with 19,926 training examples and 2,214 test examples in each fold. Before training and testing, the features were normalized using L2 normalization.

5.2 Results

We measured the performance of our system with accuracy. Table 5 presents the results broken down by fold. We also looked at our 14 advice features to see whether they correlated with the ranks. Although no single feature showed a significant linear correlation with ranks (as measured by Pearson coefficient), there are small differences in their mean and median values between ranks. Table 6 shows the comparison of raw feature values across the three ranks. We included more decimal points in the table to better reflect the differences. Table 7 presents statistical significance scores of the differences between feature values across rank pairs.

Rank		aptitude	attention	pleasantness	sensitivity	Relatability score	Imperative score	Advice score
0	Mean	0.119163	**0.087951**	0.087890	0.051194	0.025531	**0.069907**	0.025881
	Median	0.128197	0.082036	0.087306	0.039533	0.009709	0.053333	0.000000
1	Mean	0.130136	0.087693	0.099144	**0.051375**	0.027642	0.066100	**0.027425**
	Median	0.141360	0.084107	0.107313	0.037422	0.012500	0.048780	0.000000
2	Mean	**0.134223**	0.082651	**0.100406**	0.050223	**0.028502**	0.066240	0.026667
	Median	0.139275	0.082451	0.095032	0.037500	0.012085	0.048780	0.000000
Rank		ASD	ASH	ASHD	TOC	CNE	CPN	LEN
0	Mean	1.000910	0.943619	0.057292	0.141257	**0.000108**	0.049593	0.354938
	Median	0.622520	0.588571	0.030000	0.000000	0.000000	0.000000	0.200000
1	Mean	**1.047904**	**0.988323**	**0.059582**	0.140745	**0.000108**	**0.052602**	**0.378932**
	Median	0.657738	0.620119	0.032348	0.000000	0.000000	0.000000	0.220000
2	Mean	1.004613	0.947719	0.056894	**0.159900**	0.000054	0.047940	0.359827
	Median	0.640000	0.602750	0.031667	0.000000	0.000000	0.000000	0.220000

Table 6: Feature values across different ranks. We bolded highest mean value for each feature.

5 Error Analysis and Discussion

For a research problem posed in this way, it was important that each of the three comment texts went through the same initial calculations. This could have been achieved by using a recurrent neural network, where each timestep – in our case, comment text – is processed by the same unit (for example GRU [Cho *et al.*, 2014] or LSTM [Hochreiter and Schmidthuber, 1997]) that has its parameters adjusted during the training process. However, our attempts at using an RNN were unsuccessful. Although the algorithm trained well (training set accuracy was usually above 0.95), these results did not generalize to the test set. Test accuracy was always around 0.33, which in this setting is random chance level. One reason for this may be that RNNs are particularly sensitive to the order of the timesteps and even shuffling the comments did not help in alleviating this issue. The network kept overfitting the training set over the course of many epochs, but then was not able to make correct predictions on the previously unseen data of the test set. With the RNN, prediction for each comment relied heavily on calculations made for the previous one(s) instead of the network looking at the triple as a group and not as a sequence. Using a convolutional network solved this problem.

Furthermore, we made some interesting observations during the training. First, the network worked only with very specific settings, namely with the Adadelta optimization algorithm and the tanh activation function. While searching for the best optimization method and activation function is routinely performed to yield the best results for the given network, the differences in accuracy between various choices were unusually large in our case. Despite repeated training, the algorithm did not converge with any other optimization algorithm and the ReLU activation function, which we initially tried instead of tanh, caused the network to get stuck in a local minimum at a high error level. Second, around epoch 700-800 error would briefly rise and then fall down again to an even lower level. The tendency can be observed in Figure 1. This temporary drop in performance is usually caused by a learning rate that is too large for the given stage of training. It can be assumed that after Adadelta adjusted the learning rate, the network performance was able to rise up again in the last epochs. Perhaps this is the reason why this particular optimization algorithm worked best in our case: other algorithms like Adam [Kingma and Ba, 2014] or SGD [Robbins and Monro, 1951] would get stuck and be unable to overcome this issue.

As is evident from results presented in Table 5, we were able to achieve very high accuracy on our task. Perhaps this was caused by the relatively big amount of data; at over 22,000 examples the network had more than enough data to learn

Feature	Ranks 0-1	Ranks 1-2	Ranks 0-2
Aptitude	**0.035**	0.422	**0.003**
Attention	0.948	0.209	0.174
Pleasantness	**0.028**	0.803	**0.014**
Sensitivity	0.956	0.720	0.761
Relatability score	**0.017**	0.354	**0.001**
Imperative score	**0.042**	0.940	**0.044**
Advice score	0.234	0.560	0.542
ASD	0.113	0.157	0.901
ASH	0.112	0.162	0.884
ASHD	0.228	0.163	0.837
TOC	0.970	0.178	0.193
CNE	1.000	0.564	0.414
CPN	0.378	0.147	0.579
LEN	**0.054**	0.111	0.682

Table 7: Statistical significance of differences between feature values across rank pairs. We bolded p values of 0.05 or lower.

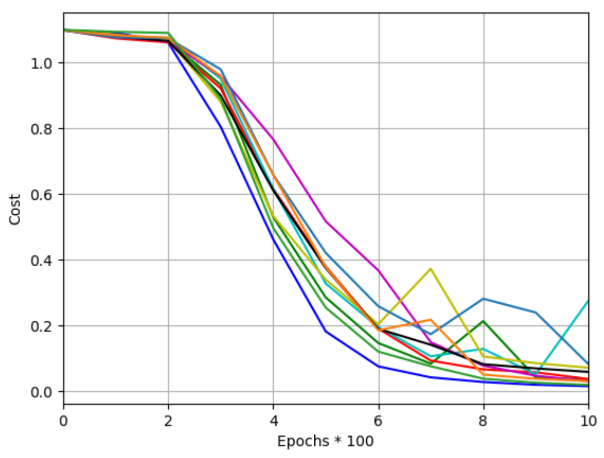

Figure 1: Overview of cost progression across 1000 training epochs for all folds. Each fold is marked with a different color.

Fold	Training loss	Training accuracy	Test loss	Test accuracy
1	0.263	0.888	0.403	0.866
2	0.347	0.866	0.375	0.840
3	0.150	0.945	0.181	0.929
4	0.353	0.849	0.439	0.828
5	0.184	0.919	0.250	0.898
6	0.190	0.943	0.211	0.932
7	0.212	0.923	0.282	0.901
8	0.207	0.920	0.245	0.903
9	0.376	0.880	0.390	0.849
10	0.248	0.923	0.287	0.899
Average	0.253	0.906	0.306	0.885

Table 8: Results of loss and accuracy values across all folds for the model trained only on word2vec features.

how to rank the comment texts. However, this task was performed on text triples, which means that the results are valid only in a very specific setting. Ideally, we would like to have a network able to rank the quality of advice contained in any given single text. However, our network specifically takes a triple of texts as input and it was not trained to recognize objective advice quality, but rather to select the best advice text from a given triple regardless of the overall quality level in that triple. Therefore, right now it cannot be used to judge how good a piece of advice is without any comparison. Constructing a network capable of accomplishing this task based on our current findings is a topic for future studies. Likewise, we assumed that all advice comments were on topic, because they were downloaded from their respective threads as responses to another user's post, but this may not always be the case with raw data obtained in a different manner. Therefore, further experiments will involve determining whether the advice is thematically appropriate for the given problem.

We performed error analysis on all the folds. First, we prepared confusion matrices to see which ranks were most commonly confused with each other. We found no clear tendency across all folds. However, we calculated mean numbers of misranked comments for all confusion matrices and we found out that the numbers were slightly higher for comments that were misranked as 2 despite the true label being 0 or 1. In other words, for comments with true label 0 there were more comments misranked as 2 than those misranked as 1, and likewise, for comments with true label 1 there were more misranked as 2 than as 0. While this result is not conclusive in any way, it shows an interesting quality about our algorithm.

We have also performed an analysis of the content of misranked comments. Since in our setup each triple of comments was present in the dataset six times, there was a possibility of each comment to be present multiple times in the test set. In such cases, comments that were misranked once tended to be misranked again on some of their subsequent appearances in the test set as well. This suggests that some single comments may be a bit troublesome for our algorithm, although the percentage of such comments in the overall dataset is negligible. Moreover, once again we found no clear characteristics of such misranked comments compared to those that were ranked correctly. This was also the case with other comments that got misranked only once. Such findings suggest that our algorithm has no defined bias in ranking, but makes mistakes randomly, which can be expected with a neural network.

Other misranked comments were of the *[deleted]* or *[removed]* kind. Comments with this kind of content were either removed by the moderators of the subreddit or deleted by the user themselves. Such comments are usually inappropriate or rude and would not receive a lot of points, which means they would usually rank the lowest in any given triple in our dataset. However, it is possible that some such comments contained content that was upvoted by people agreeing with the rude or inappropriate message and at the time of our downloading the data this comment had a high score despite being already deleted or removed. It is also possible that some user shared their advice in the comment and that advice was good enough to get a lot of upvotes, but then was deleted from discussion by the user themselves because they decided it revealed too much about them after all. This is an occasional occurrence not only in the advice subreddits, but also in subreddits concerning other personal issues, for example mental health. Whatever the cause, the *[deleted]* and *[removed]* comments were misleading to our algorithm, as the features were calculated not from the original content of the comment (which was no longer available), but from the single words *deleted* and *removed* respectively. As a result, the features were not informative enough to perform the ranking correctly. We did foresee the problems that such comment might pose when gathering data, but removing any of them from the dataset would result in removing the entire triple, which we wanted to avoid. Moreover, the *[deleted]* and *[removed]* comments were only a small fraction of our misranked comments from the test set. They most likely did not hinder the training process either, as neural networks are rather robust against occasional noise in data.

It must be noted here that the algorithm performed the final ranking by taking *argmax* of a 3x3 matrix with one-hot vector rows, so the rankings were not interchangeable, but independent at this point. In other words, a misranked comment in the triple did not translate to another comment being misranked by exchanging their mutual rankings. This means that even if a triple contained a misranked comment, other comments in that triple could be (and usually were) ranked correctly.

As can be seen in Tables 6 and 7, the differences in feature values between ranks were relatively small. For many features, like Advice Score or Specificity Scores, those differences were not statistically significant. This suggests that perhaps the network would not be able to rank advice texts based on these features alone and that it benefitted from the word2vec features as well.

Following up on these findings, we conducted additional experiments using solely word2vec features to see how much impact our 14 advice features had on the algorithm. We slightly adjusted the network architecture to accommodate the new input shape, which was (m, 1, 1, 300) instead of (m, 1, 1, 342). Therefore, the *Conv1* layer had 114 filters of shape (1, 100) instead of (1, 114). After that point, the input/output dimensions and further layers remained the same as in our models for all features. We trained the model with exactly the same hyperparameters and exactly the same number of epochs, which was 1000. The results of these experiments can be seen in Table 8. The average test accuracy was only 0.89 as compared to 0.97 from Table 5. This proves that even though word2vec features were important in our study, our 14 advice features also played a significant role in achieving good accuracy in the experiments.

We were also able to identify most important features in our study: aptitude, pleasantness, Relatability Score and Imperative Score, since the differences in their values between ranks were statistically significant. For the sentics, they are associated with dichotomies such as ecstasy-grief for pleasantness and admiration-loathing for aptitude. Our findings suggest that these emotions may be more important in advice texts than others like vigilance-amazement associated with attention or rage-terror associated with sensitivity. A lot of posts in our dataset described the author's dissatisfaction with their current life and desire to change and be happier. Because of this, emotions such as fear, anger or anticipation may have been less present in the advice comments, while talk about sadness, trust or joy seemed to be more prominent, especially when it comes to motivational advice. The statistical significance of Imperative Score and Relatability Score is noteworthy as well; we designed these features to reflect the fact that best rated advice comments tended to contain a lot of imperative expressions and were authored by people who related to the given problem. On the other hand, lack of significant differences between ranks in Advice Score is not surprising; all comments contained some form of advice, so obviously advice expressions were present in all of them. It seems that, rather than the presence of advice expressions, the manner of giving advice was more significant, specifically how many first-person pronouns and imperative phrases were included in the text. As can be seen in Table 5, best ranked comments had the highest Imperative Score and lowest rated comments had the highest Relatability Score. This suggests that while it is important to use imperative expressions when giving advice and to relate to the given problem, too much self-talk deducts from that advice's quality.

Finally, the data from Table 5 sheds a new light on previous findings concerning the features. Error analysis of experiments conducted in [Swieczkowska *et al.*, 2018] revealed differences in feature values between texts containing advice and regular ones. For almost all advice features except Average Semantic Height (which at that point was calculated differently than described here), they had perceptibly higher values for advice texts than for regular texts. This is why they could be used for a classification task mentioned in the Introduction section. We assumed that similarly, the values for advice features would be higher for good quality advice compared to lower quality advice. However, the difference in features between rank 0 and rank 2 is small. Interestingly, some features are highest for the middle rank 1, for example sensitivity, Advice Score or ASHD. All these findings suggest that the relationship between these features and advice quality is complicated and not readily visible, which is in line with our findings about the lack of linear correlations between any given feature and advice rank.

7 Conclusions

In this paper we have presented a convolutional neural network able to rank online comments containing advice based on advice quality, as judged by other online users. While this method cannot be used to determine objective quality of a piece of advice, it is useful for selecting the best advice in a given group of texts. This can be useful in creating a motivational dialogue system, for example by choosing the best advice from three candidate outputs from the system and presenting that advice to the user as the final output. We were also able to identify specific measurable qualities of a good advice text, such as scoring high on aptitude, pleasantness and Imperative Score while maintaining Relatability Score on a lower level.

Acknowledgements

The authors of this paper would like to thank prof. Hiroyuki Iizuka of Hokkaido University for his invaluable advice on this project. This work was supported by JSPS KAKENHI Grant Number 17K00295.

References

[Badubi, 2017] Reuben M. Badubi. Theories of motivation and their application in organizations: A risk analysis. *International Journal of Innovation and Economic Development*, 3(3):43–50, 2017.

[Callejas and Griol, 2016] Zoraida Callejas and David Griol. An affective utility model of user motivation for counselling dialogue systems. In *International Workshop on Future and Emerging Trends in Language Technology*, pages 86–97, 2016. Springer.

[Cambria *et al.*, 2018] Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1975–1802, 2018.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[Deshpande *et al.*, 2010] Shailesh Deshpande, Girish Keshav Palshikar, and G. Athiappan. An unsupervised approach to sentence classification. In *International Conference on Management of Data COMAD*, page 88, 2010.

[Elmelid *et al.*, 2015] Andrea Elmelid, Andrew Stickley, Frank Lindblad, Mary Schwab-Stone, Christopher C Henrich, and Vladislav Ruchkin. Depressive symptoms, anxiety and academic motivation in youth: Do schools and families make a difference? *Journal of adolescence*, 45:174–182, 2015.

[Fang *et al.*, 2017] Changjian Fang, Dejun Mu, Zhenghong Deng, and Zhiang Wu. Word-sentence co-ranking for automatic extractive text summarization. *Expert Systems with Applications*, 72:189–195, 2017.

[Fussner *et al.*, 2018] Lauren M. Fussner, Kathryn J. Mancini, and Aaron M. Luebbe. Depression and approach motivation: Differential relations to monetary, social, and food reward. *Journal of Psychopathology and Behavioral Assessment*, 40(1):117–129, 2018.

[Gerhart and Fang, 2015] Barry Gerhart and Meiyu Fang. Pay, intrinsic motivation, extrinsic motivation, performance, and creativity in the workplace: Revisiting long-held beliefs. *Annual Review of Organizational Psychology and Organizational Behavior*, 2(1):489–521, 2015.

[He *et al.*, 2010] Helen Ai He, Saul Greenberg, and Elaine M. Huang. One size does not fit all: Applying the transtheoretical model to energy feedback technology design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 927–936, 2010. ACM.

[Hershenberg, 2017] Rachel Hershenberg. *Activating happiness: A jump-start guide to overcoming low motivation, depression, or just feeling stuck*. New Harbinger Publications, 2017.

[Hochreiter and Schmidthuber, 1997] Sepp Hochreiter and Jürgen Schmidthuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Jabri *et al.*, 2018] Siham Jabri, Azzeddine Dahbi, Taoufiq Gadi, and Abdelhak Bassir. Ranking of text documents using tf-idf weighting and association rules mining. In *Proceedings of 4th International Conference on Optimization and Applications (ICOA)*, pages 1–6, 2018. IEEE.

[Kaptein *et al.*, 2012] Maurits Kaptein, Boris De Ruyter, Panos Markopoulos, and Emile Aarts. Adaptive persuasive systems: A study of tailored persuasive text messages to reduce snacking. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(2):10, 2012.

[Kaptein *et al.*, 2009] Maurits Kaptein, Panos Markopoulos, Boris de Ruyter, and Emile Aarts. Can you be persuaded? Individual differences in susceptibility to persuasion. In *Proceedings of IFIP Conference on Human-Computer Interaction*, pages 115–118, 2009. Springer.

[Kingma and Ba, 2014] Diederick Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[Litalien *et al.*, 2015] David Litalien, Frederic Guay, and Alexandre J. Morin. Motivation for PhD studies: Scale development and validation. *Learning and Individual Differences*, 41:1–13, 2015.

[Malhotra *et al.*, 2018] Dheeraj Malhotra, Monica Malhotra, and O. P. Rishi. An innovative approach of web page ranking using Hadoop and MapReduce-based cloud framework. In *Proceedings of Big Data Analytics*, pages 421–427, 2018. Springer.

[Myangah and Rezai, 2016] Tayebeh Mosavi Myangah and Mohammad Javad Rezai. Persian text ranking using lexical richness indicators. *Glottometrics*, 35:6–15, 2016.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951:400–407, 1951.

[Schneider and Kummert, 2016] Sebastian Schneider and Franz Kummert. Motivational effects of acknowledging feedback from a socially assistive robot. In Proceedings of International Conference on Social Robotics, pages 870–879, 2016. Springer.

[Sethi and Dixit, 2019] Shilpa Sethi and Ashutosh Dixit. A novel page ranking mechanism based on user browsing patterns. In Proceedings of Software Engineering, pages 37–49, 2019. Springer.

[Süssenbach *et al.*, 2014] Luise Süssenbach, Nina Riether, Sebastian Schneider, Ingmar Berger, Franz Kummert, Ingo Lütkebohle, and Karola Pitsch. A robot as a fitness companion: Towards and interactive action-based motivation model. In *Proceedings of The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 286–293, 2014. IEEE.

[Swieczkowska *et al.*, 2017] Patrycja Swieczkowska, Jolanta Bachan, Rafal Rzepka, and Kenji Araki. Asystent – A prototype of a motivating electronic assistant. In *Proceedings of the Linguistic And Cognitive Approaches To Dialog Agents (LaCATODA 2017)*, pages 11–19, 2017. CEUR Workshop Proceedings.

[Swieczkowska *et al.*, 2018] Patrycja Swieczkowska, Rafal Rzepka, and Kenji Araki. Analyzing motivation techniques in emotionally intelligent dialogue systems. In

Proceedings of Biologically Inspired Cognitive Architectures Meeting, pages 355–360, 2018. Springer, Cham.

[Vajjala and Meurers, 2016] Sowmya Vajjala and Detmar Meurers. Readability-based sentence ranking for evaluating text simplification. arXiv preprint arXiv: 1603.06009, 2016.

[Zeiler, 2012] Matthew Zeiler. ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.