Categorical Metadata Representation for Customized Text Classification

Jihyeok Kim*[1] Reinald Kim Amplayo*[2]
Kyungjae Lee[1] Sua Sung[1] Minji Seo[1] Seung-won Hwang[1]
(* equal contribution)

[1]Yonsei University [2]University of Edinburgh
zizi1532@yonsei.ac.kr reinald.kim@ed.ac.uk
{lkj0509,dormouse,ggatalminji,seungwonh}@yonsei.ac.kr

Abstract

The performance of text classification has improved tremendously using intelligently engineered neural-based models, especially those injecting categorical metadata as additional information, e.g., using user/product information for sentiment classification. This information has been used to modify parts of the model (e.g., word embeddings, attention mechanisms) such that results can be customized according to the metadata. We observe that current representation methods for categorical metadata, which are devised for human consumption, are not as effective as claimed in popular classification methods, outperformed even by simple concatenation of categorical features in the final layer of the sentence encoder. We conjecture that categorical features are harder to represent for machine use, as available context only indirectly describes the category, and even such context is often scarce (for tail category). To this end, we propose using basis vectors to effectively incorporate categorical metadata on various parts of a neural-based model. This additionally decreases the number of parameters dramatically, especially when the number of categorical features is large. Extensive experiments on various data sets with different properties are performed and show that through our method, we can represent categorical metadata more effectively to customize parts of the model, including unexplored ones, and increase the performance of the model greatly.

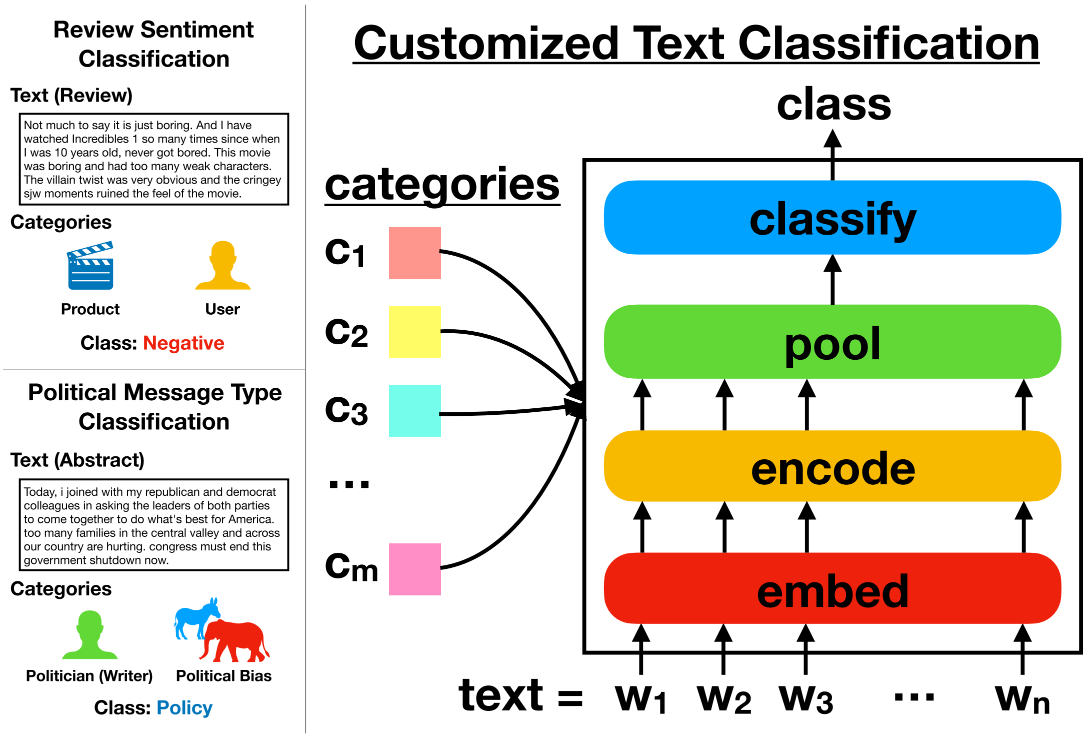
1 Introduction

Text classification is the backbone of most NLP tasks: review classification in sentiment analysis (Pang et al., 2002), paper classification in scientific data discovery (Sebastiani, 2002), and question classification in question answering (Li and Roth, 2002), to name a few. While prior methods require intensive feature engineering, recent methods enjoy automatic extraction of features from text using neural-based models (Socher et al., 2011) by encoding texts into low-dimensional dense feature vectors.

This paper discusses **customized text classification**, generalized from personalized text classification (Baruzzo et al., 2009), where we customize classifiers based on possibly multiple different known categorical metadata information (e.g., user/product information for sentiment classification) instead of just the user information. As shown in Figure 1, in addition to the text, a customizable text classifier is given a list of categories specific to the text to predict its class. Existing works applied metadata information to improve the performance of a model, such as user and product (Tang et al., 2015) information in sentiment classification, and author (Rosen-Zvi et al., 2004) and publication (Joorabchi and Mahdi, 2011) information in paper classification.

Towards our goal, we are inspired by the advancement in neural-based models, incorporating categorical information "as is" and injecting it on various parts of the model such as in the word embeddings (Tang et al., 2015), attention mechanism (Chen et al., 2016; Amplayo et al., 2018a) and memory networks (Dou, 2017). Indeed, these methods theoretically make use of combined features from both textual and categorical features, which make them more powerful than disconnected features. However, metadata is generated for human understanding, and thus we claim that these categories need to be carefully represented for machine use to

Transactions of the Association for Computational Linguistics, vol. 7, pp. 201–215, 2019. Action Editor: Bo Pang.
Submission batch: 11/2018; Revision batch: 1/2019; Final submission: 2/2019; Published 4/2019.


Review Sentiment Classification
Text (Review)

Not much to say it is just boring. And I have watched Incredibles 1 so many times since when I was 10 years old, never got bored. This movie was boring and had too many weak characters. The villain twist was very obvious and the cringey sjw moments ruined the feel of the movie.

Categories
Product
User
Class: Negative
Political Message Type Classification
Text (Abstract)

Today, i joined with my republican and democrat colleagues in asking the leaders of both parties to come together to do what's best for America. too many families in the central valley and across our country are hurting. congress must end this government shutdown now.

Figure 1: A high-level framework of models for the Customized Text Classification Task that inputs a text with n tokens (e.g., review) and m categories (e.g., users, products) and outputs a class (e.g., positive/negative). Example tasks are shown in the left of the figure.

improve the performance of the text classifier effectively.

First, we empirically invalidate the results from previous studies by showing in our experiments on multiple data sets that popular methods using metadata categories "as is" perform worse than a simple concatenation of textual and categorical feature vectors. We argue that this is because of the difficulties of the model in learning optimized dense vector representation of the categorical features to be used by the classification model. The reasons are two-fold: (a) categorical features do not have direct context and thus rely solely on classification labels when training the feature vectors, and (b) there are categorical information that are sparse and thus cannot effectively learn optimal feature vectors.

Second, we suggest an alternative representation, using low-dimensional basis vectors to mitigate the optimization problems of categorical feature vectors. Basis vectors have nice properties that can solve the issues presented here because they (a) transform multiple categories into useful combinations, which serve as mutual context to all categories, and (b) intelligently initialize vectors, especially of sparse categorical information, to a suboptimal location to efficiently train them further. Furthermore, our method reduces the number of trainable parameters and thus is flexible for any kinds and any number of available categories.

We experiment on multiple classification tasks with different properties and kinds of categories available. Our experiments show that while customization methods using categorical information "as is" do not perform as well as the naive concatenation method, applying our proposed basis-customization method makes them much more effective than the naive method. Our method also enables the use of categorical metadata to customize other parts of the model, such as the encoder weights, that are previously unexplored due to their high space complexity and weak performance. We show that this unexplored use of customization outperform popular and conventional methods such as attention mechanism when our proposed basis-customization method is used.

2 Preliminaries

2.1 Problem: Customized Text Classification

The original text classification task is defined as follows: Given a text $W = \{w_1, w_2, ..., w_n\}$, we are tasked to train a mapping function $f(W)$ to predict a correct class $y \in \{y_1, y_2, ..., y_p\}$ among the p classes. The **customized text classification** task makes use of the categorical metadata information attached on the text to customize the mapping function. In this paper, we define categorical metadata as non-continuous information that describes the text.[1] An example task is review sentiment classification with user and product information as categorical metadata.

Formally, given a text $t = \{W, C\}$, where $W = \{w_1, w_2, ..., w_n\}$, $C = \{c_1, c_2, ..., c_m\}$, w_x is the xth of the n tokens in the text, and c_z is the category label of the text on the zth category of the m available categories, the goal of customized text classification is to optimize a function $f_C(W)$ to predict a label y, where $f_C(W)$ is the classifier dependent with C. In our example task, W is the review text, and we have $m = 2$ categories where c_1 and c_2 are the user and product information.

This is an interesting problem because of the vast opportunities it provides. First, we are motivated to use categorical metadata because existing work has shown that non-textual additional information, such as POS tags (Go et al., 2009) and latent topics (Zhao et al., 2017), can be used as strong supplementary supervision to improve the performance of text classification. Second, while previously used additional information is found to be helpful, they are either domain-dependent or very noisy (Amplayo et al., 2018b). On the other hand, categorical metadata are usually factual and valid information that are either inherent (e.g., user/product information) or human-labeled (e.g., research area). Finally, the customized text classification task generalizes the personalization problem (Baruzzo et al., 2009), where instead of personalizing based on single user information, we *customize* based on possibly multiple categories, which may or may not include user information. This consequently creates an opportunity to develop customizable virtual assistants (Papacharissi, 2002).

2.2 Base Classifier: BiLSTM

We use a Bidirectional Long Short Term Memory (BiLSTM) network (Hochreiter and Schmidhuber, 1997) as our base text classifier as it is proven to work well on classifying text sequences (Zhou et al., 2016). Although the methods that are described here apply to other effective classifiers as well, such as convolutional neural networks (CNNs) (Kim, 2014) and hierarchical models (Yang et al., 2016), we limit our experiments to BiLSTM to cover more important findings.

Our BiLSTM classifier starts by encoding the word embeddings using a forward and a backward LSTM. The resulting pairs of vectors are concatenated to get the final encoded word vectors, as shown here:

$$w_i \in \mathbb{W} \tag{1}$$

$$\overrightarrow{h}_i = LSTM_f(w_i, \overrightarrow{h}_{i-1}) \tag{2}$$

$$\overleftarrow{h}_i = LSTM_b(w_i, \overleftarrow{h}_{i+1}) \tag{3}$$

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i] \tag{4}$$

Next, we pool the encoded word vectors h_i into a text vector d using an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015), which calculates importance scores using a latent context vector x for all words, normalizes the scores using softmax, and uses them to do weighted sum on encoded word vectors, as shown:

$$e_i = x^\top h_i \tag{5}$$

$$a_i = \frac{exp(e_i)}{\sum_j exp(e_j)} \tag{6}$$

$$d = \sum_i h_i * a_i \tag{7}$$

Finally, we use a logistic regression classifier to classify labels using learned weight matrix $W^{(c)}$ and bias vector $b^{(c)}$:

$$y' = W^{(c)} d + b^{(c)} \tag{8}$$

We can then train our classifier using any gradient descent algorithm by minimizing the negative log likelihood of the log softmax of predicted labels y' with respect to the actual labels y.

[1] We limit our scope to texts with categorical metadata information (product reviews, news articles, tweets, etc.), which covers most of the texts on the Web. Texts without metadata can use predicted categorical information, such as topics from a topic model, which are commonly used (Zhao et al., 2017; Chou et al., 2017). However, because the prediction may be incorrect, performance gains cannot be guaranteed. We leave the investigation of this area in future work.

2.3 Baseline 1: Concatenated BiLSTM

To incorporate the categories into the classifier, a simple and naive method is to concatenate the categorical features with the text vector d. To do this, we create embedding spaces for the different categories and get the category vectors $c_1, c_2, ..., c_m$ based on the category labels of text d. We then use the concatenated vector as features for the logistic regression classifier:

$$y' = W^{(c)}[d; c_1; c_2; ...; c_m] + b^{(c)} \tag{9}$$

2.4 Baseline 2: Customized BiLSTM

Although the Concatenated BiLSTM easily makes use of the categories as additional features for the classifier, it is not able to leverage on the possible low-level dependencies between textual and categorical features.

There are different levels of dependencies between texts and categories. For example, when predicting the sentiment of a review "*The food is very sweet*," given the user who wrote the review, the classifier should give a positive label if the user likes sweet foods and a negative label otherwise. In this case, the dependency between the review and the user is on the *higher level*, where we look at relationships between the full text and the categories. Another example is when predicting the acceptance of a research paper given that the research area is NLP, the classifier should focus more on NLP words (e.g., language, text) rather than less-related words (e.g., biology, chemistry). In this case, the dependency between the research paper and the research area is on the *lower level*, where we look at relationships between segments of text and the categories.

We present five levels of Customized BiLSTM, which differ on the location where we inject the categorical features, listed here from the highest level to the lowest level of dependencies between text and categories. The main idea is to impose category-specific weights, rather than a single weight at each level of the model:

1. **Customize on the bias vector**: At this level of customization, we look at the **general biases** the categories have towards the problem. As a concrete example, when classifying the type of message a politician wrote, he/she can be biased towards writing personal messages than policy messages. Instead of using a single bias vector $b^{(c)}$ in the logistic regression classifier (Equation 8), we use additional multiple bias vectors for each category, as shown below. In fact, this is in spirit essentially equivalent to concatenated BiLSTM (Equation 9), where the derivation is:

$$\begin{aligned} y' &= W_d d + b_{c_1} + ... + b_{c_m} + b^{(c)} \\ &= W_d d + W_{c_1} c_1 + ... + W_{c_m} c_m + b^{(c)} \\ &= W^{(c)}[d; c_1; c_2; ...; c_m] + b^{(c)} \end{aligned}$$

2. **Customize on the linear transformation**: At this level of customization, we look at the **text-level semantic biases** the categories have. As a concrete example, in the sentiment classification task, the review "*The food is very sweet*" can have a negative sentiment if the user who wrote the review does not like sweets. Instead of using a single weight matrix $W^{(c)}$ in the logistic regression classifier (Equation 8), we use different weight matrices for each category:

$$y' = W^{(c)}_{c_1} d + W^{(c)}_{c_2} d + ... + W^{(c)}_{c_m} d + b^{(c)}$$

3. **Customize on the attention pooling**: At this level of customization, we look at the **word importance biases** the categories have. A concrete example is, when classifying a research paper, NLP words should be focused more when the research area is NLP. Instead of using a single context vector x when calculating the attention scores e (Equation 5), we use different context vectors for each category:

$$\begin{aligned} e_i &= x_{c_1}^\top h_i + x_{c_2}^\top h_i + ... + x_{c_m}^\top h_i \\ a &= softmax(e) \\ d &= \sum_i h_i * a_i \end{aligned}$$

4. **Customize on the encoder weights**: At this level of customization, we look at the **word contextualization biases** the categories need. A concrete example is, given the text ''*deep learning for political message classification*'', when encoding the word *classification*, the BiLSTM should retain the semantics of words *political message* more and forget the semantics of other words more when the research area is about politics. Instead of

using a single set of input, forget, output, and memory cell weights for each LSTM (Equations 2 and 3), we use multiple sets of the weights, one for each category:

$$\begin{bmatrix} g_t \\ i_t \\ f_t \\ o_t \end{bmatrix} = \begin{bmatrix} tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\sum_{0<k\leq m} W^{(e)}_{c_k}[w_t; h_{t-1}] + b \right)$$

5. **Customize on the word embeddings**: At this level of customization, we look at the **word preference biases** the categories have. For example, a user can prefer the use of word ''*terribly*'' as a positive adverb rather than the more common usage of the word with negative sentiment. Instead of directly using the word vectors from the embedding space $\mathbb{W}$ (Equation 1), we add a residual vector calculated based on a nonlinear transformation of the word vector using category-specific weights:

$$r = tanh(W^{(w)}_{c_1} w_i + ... + W^{(w)}_{c_m} w_i) \quad (10)$$
$$w_i = w_i + r$$

Previous work has proposed customization on bias vectors and word embeddings (Tang et al., 2015), and on attention pooling (Chen et al., 2016). We are the first to introduce customization on the linear transformation matrix and the encoders. Moreover, we are the first to use residual perturbations as word meaning modification for customizing word embeddings, in which we saw better performance than using a naive affine transformation, proposed in Tang et al. (2015), in our prior experiments.

3 Proposed Method

3.1 Problems of Customized BiLSTM

As explained in the previous section, Customized BiLSTM should perform better than Concatenated BiLSTM. However, that is only if the optimization of category-specific weights operates properly for machine usage. Training the model to optimize these weights is very difficult for two reasons.

First, categorical information has unique properties that make it nontrivial to train. One property is that unlike texts that naturally use neighboring words/sentences as context (Lin et al., 2015; Peters et al., 2018), categorical information stands alone and thus does not have information aside from itself. This forces the learning algorithm to rely solely on the classification labels y to find the optimal category-specific weights. Another property is that some categories may contain labels that are sparse or do not have enough instances. For example, a user can be cold-start (Lam et al., 2008) or does not have enough reviews. In this case, the problem expands to few-shot learning (Li et al., 2006). Thus weights are hard to optimize using gradient-based techniques (Ravi and Larochelle, 2016).

Second, the number of weights is multiplied by the number of categories m and the number of category labels each category has, which enlarges the number of parameters needed to be trained as m increases. This magnifies the problems of context absence and information sparsity described above, since optimizing large parameters with limited inductive bias is very difficult. Moreover, because of the large parameters, some methods may not fit in commercially available machines and thus may not be practically trainable.

3.2 Basis Customization

We propose to solve these problems by using basis vectors to produce basis-customized weights, as shown visually in Figure 2. Specifically, we use a trainable set of $d \ll dim$ basis vectors $B = \{b_1, b_2, ..., b_d\}$, where dim is the dimension of the original weights. Let V_c be the vector search space that contains all the optimal customized weight vectors v_c, such that B is the basis of V_c. Basis vectors follow the spanning property, thus we can represent all vectors in $v \in V_c$ as a linear combination of B—that is $v_c = \sum_i \gamma_i * b_i$, where the γs are the coefficients. Moreover, because we set d to a small number, we constrain the search space to a smaller vector space. Hence we can find the optimal weights in a constrained search space much faster.

To determine the γ coefficients, we first set the concatenated category vectors of the text $q = [c_1; c_2; ...; c_m]$ as the query vector, and use a trainable set of key vectors $K = \{k_1, k_2, ..., k_d\}$. We then calculate the dot product between the query and key vectors, and finally use softmax to create γ coefficients that sum to one:

$$z_i = q^\top k_i$$
$$\gamma_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

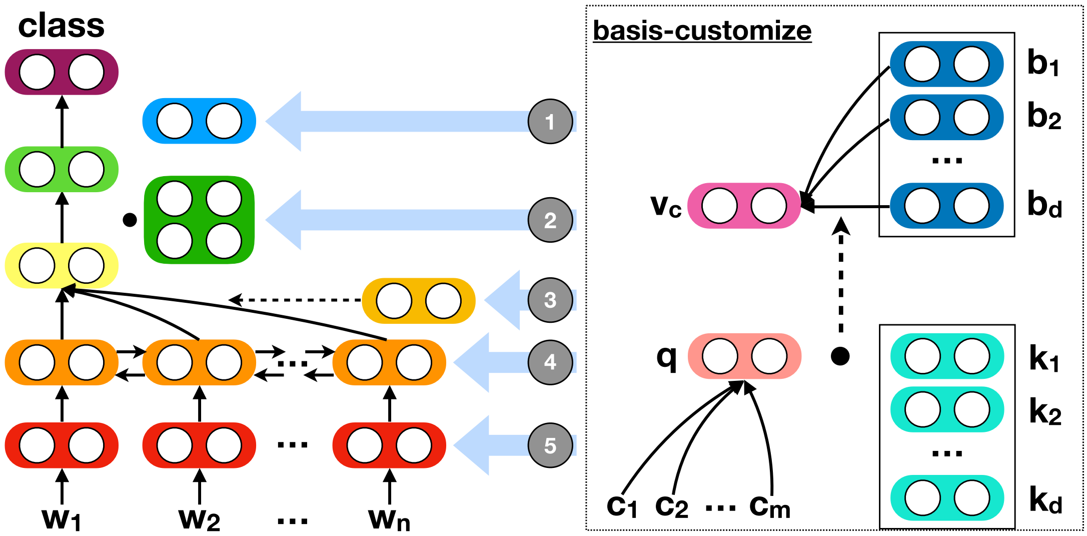


Figure 2: The full architecture of the proposed model, basis-customizing parts of the BiLSTM model: (1) the bias vector, (2) the linear transformation matrix, (3) the attention context vector, (4) the BiLSTM encoder weights, and (5) the word embeddings.

We can then use the γ coefficients to basis-customize a specific weight v, namely, $v_c = \sum_i \gamma_i * b_i$. In our BiLSTM classifier, we can basis-customize one of the following weights: (1) the bias vector $v = b^{(c)}$ and (2) the linear transformation matrix $v = W^{(c)}$ of the logistic regression classifier in Equation 8, (3) the context vector $v = x$ of the attention mechanism in Equation 5, (4) the BiLSTM weights $v = W^{(e)}$ in Equations 2 and 3, and (5) the nonlinear transformation matrix $v = W^{(w)}$ on the residual vector in Equation 10 to modify the word embeddings. These correspond to the five versions of Customized BiLSTM discussed earlier.

Basis-customizing weights help solve the problems of customizing BiLSTM in three ways. First, the basis vectors serve as fuzzy clusters of all the categories, that is, we can say that two sets of category labels are similar if they have similar γ coefficients. This information can serve as mutual context information that helps the learning algorithm find optimal weights. Second, because the search space V_c is constrained, the model is forced to initialize the category vectors and look for the optimal vectors inside the constrained space. This smart initialization contributes to situate vectors of sparse categorical information to a sub-optimal location and efficiently trains them further, despite the lack of instances. Finally, because we only use a very small set of basis vectors, we reduce the number of weights dramatically.

4 Experiments

We experiment on three data sets for different tasks: (1) the Yelp 2013 data set[2] (Tang et al., 2015) for Review Sentiment Classification, (2) the AAPR data set[3] (Yang et al., 2018) for Paper Acceptance Classification, and (3) the PolMed data set[4] for Political Message Type Classification. Statistics, categories, and properties of the data sets are reported in Table 1. Details about the data sets are discussed in the next sections.

General experimental settings are as follows. The dimensions of the word vectors are set to 300. We use pre-trained GloVe embeddings (Pennington et al., 2014) to initialize our word vectors. We create UNK tokens by transforming tokens with frequency less than five into UNK. We handle unknown category labels by setting their corresponding vectors to zero. We tune the number of basis vectors d using a development set, first by sweeping across 2 to 30 with large intervals, and then by searching through the neighbors of the best configuration during the first sweep. Interestingly, d tends to be very small, between values 2 to 4. We set the batch size to 32. We use stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler,

[2] http://ir.hit.edu.cn/~dytang.
[3] https://github.com/lancopku/AAPR.
[4] https://www.figure-eight.com/.

Data Set	Splits	Categories	Properties
Yelp 2013	62,522 / 7,773 / 8,671	• users (1.6k) • products (1.6k)	Categories can be sparse (i.e., there may not be enough reviews for each user/product).
AAPR	33,464 / 2,000 / 2,000	• authors (48k) • research area (144)	Authors are sparse and have many category labels. Categories can have multiple labels (e.g., multiple authors, multidisciplinary fields).
PolMed	4,500 / 0 / 500	• politician (505) • media source (2) • audience (2) • political bias (2)	The data set has more categories. Categories with binary labels may not be diverse enough to be useful.

Table 1: The data sets, the split sizes (train, dev, test), and the available categories and their properties. Numbers inside parentheses are the number of unique category labels.

Models		Yelp 2013			AAPR		PolMed	
		Accuracy	RMSE	Param	Accuracy	Param	Accuracy	Param
Base: BiLSTM		63.7	0.687	442k	61.70	188k	40.30	86k
bias vector (concat)	cust	66.3	0.661	1.3m	65.30	6.3m	40.57	121k
	basis-cust	66.9	0.654	653k	64.80	1.7m	40.92	95k
linear trasformation*	cust	59.6	0.758	4.6m	63.55	6.3m	40.04	379k
	basis-cust	**67.1**	**0.662**	655k	**65.75**	1.7m	**41.89**	96k
attention pooling	cust	65.4	0.674	1.3m	62.80	6.3m	40.93	119k
	basis-cust	66.0	0.671	652k	**65.85**	1.7m	41.73	95k
encoder weights*	cust	-	-	-	-	-	40.26	43.5m
	basis-cust	**66.1**	**0.665**	1.5m	**66.15**	2.1m	**41.42**	179k
word embedding*	cust	58.4	0.767	294m	-	-	40.84	46.0m
	basis-cust	**66.1**	**0.666**	1.0m	**65.80**	2.0m	41.58	455k

Table 2: Accuracy, RMSE, and parameter values of competing models for all data sets. An asterisk (*) indicates customization methods first introduced in this paper. A dash (-) indicates the model is too big to be trained in an NVIDIA 1080 Ti GPU. **Boldface** indicates that the performance of basis-customization is significantly better ($p < 0.05$) than that of a simple customization. Values colored red are performance weaker than that of the BiLSTM model, thus customization hurts the performance in those cases.

2012) with l_2 constraint of 3. We do early stopping using the accuracy of the development set. We perform 10-fold cross-validation on the training set when the development set is not available. Data set–specific settings are described in their corresponding sections.

We compare the performance of the following competing models: the base classifier BiLSTM with no customization, the five versions (i.e., bias, linear, attention, encoder, embedding) of Customized BiLSTM, and our proposed basis-customized versions. We report the accuracy and the number of parameters of all models, and additionally report the root mean square error (RMSE) values for the sentiment classification task. We also compare with results from previous papers whenever available. Results are shown in Table 2, and further discussion is provided the following sections.

4.1 Review Sentiment Classification

Review sentiment classification is a task of predicting the sentiment label (e.g., 1 to 5 stars) of a review text (Pang et al., 2002). We use users and products as categorical metadata. One main characteristic of the categorical information here is that both user and product can be cold-start entities (Amplayo et al., 2018a). Thus issues on sparseness may aggravate. We use 256 dimensions for the hidden states in the BiLSTM encoder and the context vector in the attention mechanism, and 64 dimensions for each of the user and product category vectors.

The results in Table 2 show that when using Customized BiLSTM, customizing on the bias vector (i.e., Concatenated BiLSTM) performs the best compared to customizing on other parts of the model with lower dependencies, which is counterintuitive and contrary to previously reported

Models		Acc	RMSE
UPNN (Tang et al., 2015)	CNN + word-cust + bias-cust	59.6	0.784
UPDMN (Dou, 2017)	LSTM + memory-cust	63.9	0.662
NSC (Chen et al., 2016)	LSTM + attention-cust	65.0	0.692
HCSC (Amplayo et al., 2018a)	BiLSTM + CNN + attention-cust (CSAA)	65.7	**0.660**
PMA (Zhu and Yang, 2017)	HierLSTM + attention-cust (PMA)	65.8	0.668
DUPMN (Long et al., 2018)	HierLSTM + memory-cust	66.2	0.667
CMA (Ma et al., 2017)	HierAttention + attention-cust (CMA)	**66.4**	0.677
Our best models	BiLSTM + encoder-basis-cust	66.1	0.665
	BiLSTM + bias-basis-cust	66.9	**0.654**
	BiLSTM + linear-basis-cust	**67.1**	0.662

Table 3: Performance comparison of previous and our best models in the Yelp 2013 dataset. Our best models perform better, even though we only use a single BiLSTM encoder. **Boldface** correspond to the best values for each block. Underlined values correspond to the best values across the board.

results. Moreover, the performance of customizing on the linear transformation matrix and word embedding is weaker than that of the base BiLSTM model, and customizing on the encoder weights makes the model too big to be trained in our GPU. When using our proposed basis-customization method, we obtain a significant increase in performance on all levels of customization in almost all performance metrics. Overall, a BiLSTM basis-customized on the linear transformation matrix, the bias vector, and the encoder weights perform the best among the models. Finally, we reduce the number of parameters dramatically by at least half compared with the Customized BiLSTM, which enables the training of Basis-Customized BiLSTM on encoder weights.

In addition to the competing models above, we also report results from previous state-of-the-art sentiment classification models that use user and product information: (a) **UPNN** (Tang et al., 2015) uses a CNN encoder and customizes on bias vectors and word embeddings; (b) **UPDMN** (Dou, 2017) uses an LSTM encoder and customizes on memory vectors; (c) **NSC** (Chen et al., 2016) uses a hierarchical LSTM encoder and customizes on attention mechanism; (d) **HCSC** (Amplayo et al., 2018a) uses a BiLSTM and a CNN as encoders and customizes on a cold-start aware attention mechanism (CSAA); (e) **PMA** (Zhu and Yang, 2017) uses a hierarchical LSTM encoder and customizes on PMA, an attention mechanism guided by external features; (f) **DUPMN** (Long et al., 2018) uses a hierarchical LSTM encoder and customizes on memory vectors; and (g) **CMA** (Ma et al., 2017) uses a hierarchical attention-based encoder and customizes on user- and product-specific attention mechanism (CMA). The comparison in Table 3 shows that our methods outperform previous models, even though (1) we only use a single BiLSTM encoder rather than more complicated ones (UPDMN and DUPMN use deep memory networks, NSC, PMA, and CMA use hierarchical encoders) and (2) we only customize on one part of the model rather than on multiple parts (UPNN customizes on bias vectors and word embeddings).

4.2 Paper Acceptance Classification

Paper acceptance classification is a task of predicting whether the paper in question is accepted or rejected (Yang et al., 2018). We use the authors[5] and the research area of the papers as categorical metadata. Both authors and research field information accept multiple labels per instance (e.g., multiple authors, multidisciplinary field), hence learning the category vector space properly is crucial to perform vector operations (Mikolov et al., 2013). We use 128 dimensions for both the hidden states in the BiLSTM encoder and the context vector in the attention mechanism and 32 dimensions for each of the categorical information. We use the paper abstract as the text. To handle multiple labels, we find that averaging the category vectors works well.

The results in Table 2 show similar trends from the sentiment classification results. First, we obtain better performance when using Concatenated BiLSTM than when using Customized BiLSTM.

[5] In reviewing scenarios, the use of authors as additional information is discouraged for fairness. We show how powerful these features are for prediction when properly modeled, which is useful for other scenarios, for example, deciding which arXiv papers to read.

Models	Accuracy
using full text (Yang et al., 2018)	
LSTM	60.5
MHCNN	**67.7**
using abstract and categories (our setting)	
LSTM	60.6
MHCNN	63.7
BiLSTM	61.7
BiLSTM+word-basis-cust	65.8
BiLSTM+attention-basis-cust	65.9
BiLSTM+encoder-basis-cust	**66.2**

Table 4: Performance comparison of models using full texts and our implemented models using paper abstracts (and authors and research areas as categories for basis-customized models) as inputs in the AAPR data set.

Second, incorporating metadata information on the attention mechanism does not perform as well as previously reported. Third, when customizing on encoder weights and word embedding, the model parameters are too big to be trained on a commercial GPU. Finally, we see significant improvements in all levels of customization when using our proposed basis-customization method, except on the bias vectors where we obtain comparable results. Overall, a BiLSTM basis-customized on the encoder weights, the attention pooling, and the word embedding perform the best among all the models. We also see at least 3.7x reduction of parameters when comparing Customized BiLSTM and Basis-Customized BiLSTM.

We also compare our results from previous literature (Yang et al., 2018), where they proposed a modular and hierarchical CNN-based encoder (MHCNN), and used the full text (i.e., from the title and authors up to the conclusion section), rather than just the abstract, the author and the research area information. Results are reported in Table 4, although full text and abstract results are not directly comparable because the original authors did not release the train/dev/test splits of their experiments. We instead re-run MHCNN using our settings and compare with our models. The results show that using either full text or abstract as input to LSTM produces similar results, thus using just the abstract can give us similar predictive bias when using the full text, at least in this data set. Moreover, our best models (1) perform significantly better ($p < 0.5$) than MHCNN when restricted to our settings, and (2) are competitive with the state-of-the-art, even though we use a simple BiLSTM encoder and only have access to the abstract, authors, and research area information.

4.3 Political Message Type Classification

Political message type classification is a task of predicting the type of information a message written by a politician is conveying, with the following nine types: attack, constituency, information, media, mobilization, personal, policy, support, and others. Two characteristics of this data set different from others are (a) that it has four kinds of categorical information: the audience (national or constituency), bias (neutral or partisan), politician, and the source (Twitter or Facebook) information, and (b) that the category types of three categories are not diverse as they only have binary category labels. Because all of these categories may not give useful information biases to the classifier, models should be able to select which categories are informative or not. We use 64 dimensions for the hidden states in the BiLSTM encoder and the context vector in the attention mechanism, and 16 dimensions for the category vectors of each of the categorical information.

The results in Table 2 also show similar trends from the previous task, but because the data set is smaller, we can compare the performance of the model when customizing on encoder weights. We show that Customized BiLSTM on linear transformation matrix and encoder weights shows weaker performance than the base BiLSTM model, Basis-Customized BiLSTM on the same levels shows significantly improved performance, and Basis-Customized BiLSTM on linear transformation matrix performs the best among the competing models. The parameters also decreased dramatically, especially on encoder weights and on word embedding where we see at least 100x difference in parameter size.

5 Analysis

5.1 Semantics of Basis Attention Vectors

We investigate how basis vectors understand word-level semantics through the lens of the attention vectors they create. Previous models either combine user/product information into a single attention vector (Chen et al., 2016) or entirely separate them into distinct user and product attention vectors (Amplayo et al., 2018a). On the other hand, our model creates a single

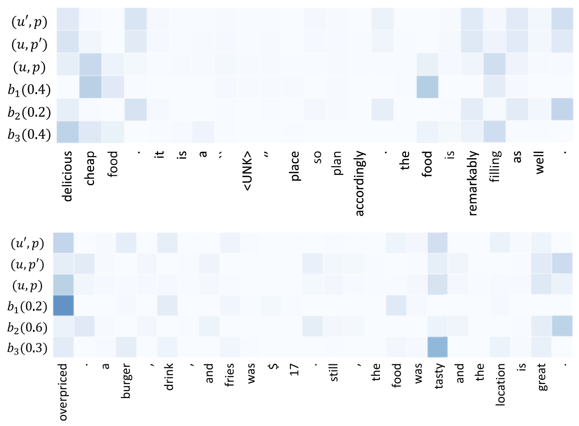


Figure 3: Examples of attention vectors from three different pairs of users and products (u',p), (u,p'), (u,p), and from the basis vectors. Numbers in parentheses are the γ_i coefficient of the pair (u,p) with respect to basis b_i.

attention vector, but through the k basis attention vectors, which are vectors containing fuzzy semantics among users and products. Figure 3 shows two examples of six attention vectors regarding a single text in the Yelp 2013 data set using the following: (1) the original user, product pair (u,p); (2–3) a sampled user/product paired with the original product/user (u',p) and (u,p'); and (4–6) the basis vectors. We can see in the first example that the first basis vector focuses on ''cheap'' and the third basis vector focuses on ''delicious.'' An interesting output is by user u, such that they wants cheaper food in product p yet care more about the taste in product p'.

5.2 Document-level Customized Dependencies

Previous literature only focused on the analysis (Amplayo et al., 2018a) and case studies (Chen et al., 2016) of word-level customized dependencies, usually through attention vectors. In this paper, we additionally investigate the document-level customized dependencies, namely, how our basis-customization changes the document-level semantics when a category is different. Table 5 shows two examples, one from the AAPR data set and one from the Political Media data set, with a variable category research area and political bias, respectively. In the first example, the abstract refers to a study on bi-sequence classification problem, a task mainly studied in the natural language processing domain, and thus is classified as accepted when the research area category is cs.CL. The model also classifies the paper as accepted when the research area is cs.IR because the two areas are related. However, when the research area is changed to an unrelated area like cs.CR, the paper is rejected. In the second example, the classifier predicts that when a politician with a neutral bias posts a Christmas greeting and mentions people who work on holidays, he is conveying a personal message. However, when the politician is biased towards a political party, the classifier thinks that the message is to offer support to those workers who are unable to be with their families.

5.3 Learning Strategy of Basis-customized Vectors

We argue that because the basis vectors B limit the search space into a constrained vector space V_c, finding the optimal values of the basis-customized vectors is faster. We show in Figure 4 the difference between the category vector space of Customized BiLSTM and of Basis-Customized BiLSTM. We see that the vector space of Customized BiLSTM looks random, with very few noticeable clusters, even when we iterate with four epochs. On the other hand, the basis-customized vector space starts as a cluster of one continuous spiral line, then starts to break down into smaller clusters. Multiple clusters of vectors in the vector space are clearly seen when the epoch is 4. Therefore, using the basis vectors makes optimization more efficient by following the learning strategy of starting from one cluster and dividing into smaller coherent clusters. This can also be shown in the visualization of the γ coefficients (also shown in the figure), where the coefficient values that are clumped together gradually spread out to their optimal values.

5.4 Performance on Sparse Conditions

We look at the performance of three models, BiLSTM, Customized BiLSTM, and Basis-Customized BiLSTM, per review frequency of user or product. Figure 5 shows plots of the accuracy of the models over different user review frequency and product review frequency on the Yelp 2013 data set. We observe that naive customization drops the performance of the BiLSTM model as the frequency of user/product review decreases. This means that the model is heavily reliant on large amounts of data for optimization. On the other hand, because basis customization can learn the optimal weights of category vectors more intelligently, it improves the performance of the model across all ranges of review frequency.

Abstract	Several tasks in argumentation mining and debating, question-answering, and natural language inference involve classifying a sequence in the context of another sequence (referred as bi-sequence classification). For several single sequence classification tasks, the current state-of-the-art approaches are based on recurrent and convolutional neural networks. On the other hand, for bi-sequence classification problems, there is not much understanding as to the best deep learning architecture. In this paper, we attempt to get an understanding of this category of problems by extensive empirical evaluation of 19 different deep learning architectures (specifically on different ways of handling context) for various problems originating in natural language processing like debating, textual entailment and question-answering. Following the empirical evaluation, we offer our insights and conclusions regarding the architectures we have considered. We also establish the first deep learning baselines for three argumentation mining tasks.		
Research Area	cs.CL (Computation and Language)	cs.IR (Information Retrieval)	cs.CR (Cryptography and Security)
Classification	Accept	Accept	Reject

Message	<UNK> christmas and happy holidays from my family to yours. wishing special <UNK> to those first responders and military personnel working to ensure our safety who are unable to be with their families this holiday season. we are all thank you for your service and dedication.	
Political Bias	Neutral	Partisan
Classification	Personal	Support

Table 5: Example texts from the AAPR data set (upper) and Political Media data set (lower) with a variable category label (research field and political bias) that changes the classification label.

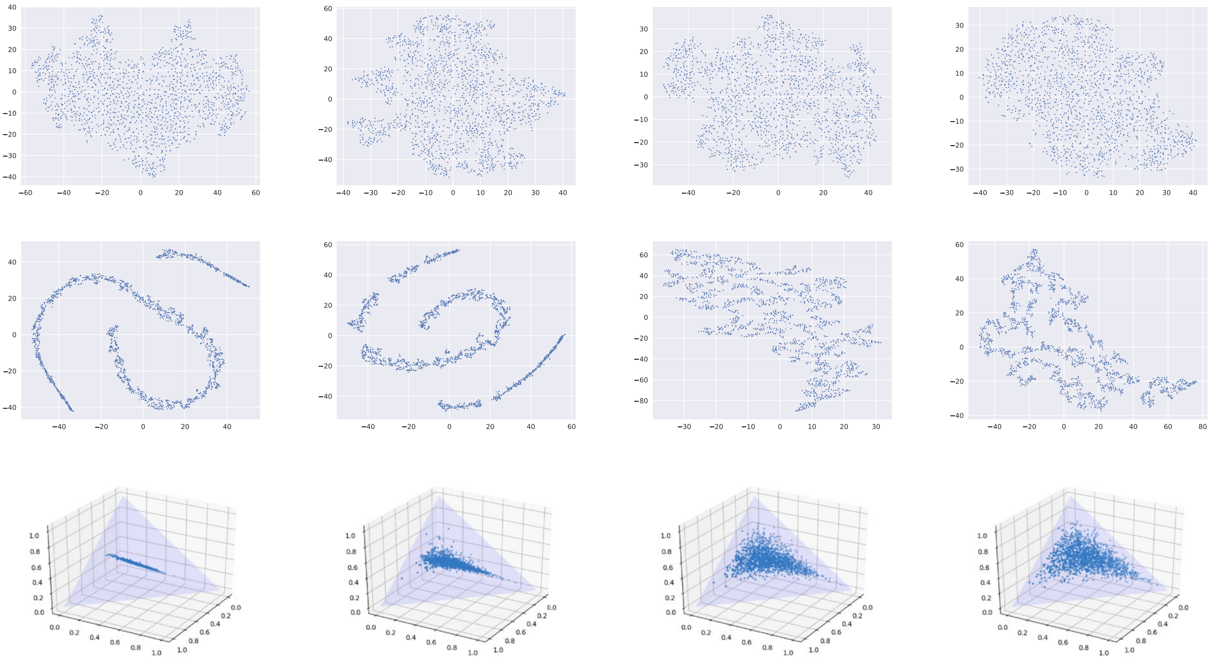


Figure 4: TSNE Visualization of the category vectors of Customized BiLSTM (first row) and Basis-Customized BiLSTM (middle row), and the γ coefficients of the latter model (last row), when epoch is equal to 1, 2, 4, and when training has finished (left to right).

We finally examine the performance of our models when data contain cold-start entities (i.e., users/products may have zero or very few reviews) using the Sparse80, subset of the Yelp 2013 data set provided in Amplayo et al. (2018a). We compare our models with three competing models: NSC (Chen et al., 2016), which uses a hierarchical LSTM encoder coupled with customization on the attention mechanism, BiLSTM+CSAA (Amplayo et al., 2018a), which uses a BiLSTM encoder with customization on a CSAA mechanism, and HCSC (Amplayo et al., 2018a), which is

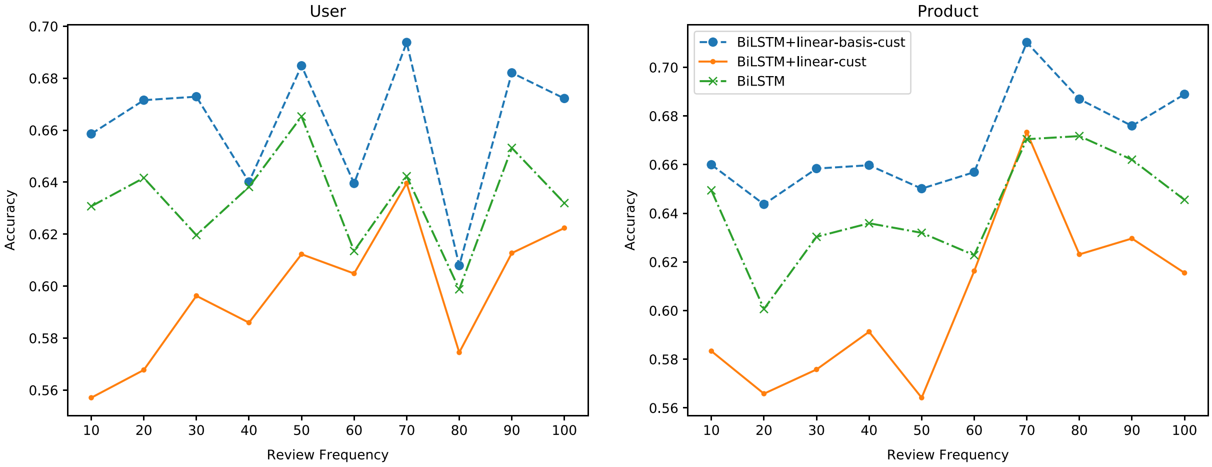


Figure 5: Accuracy per user/product review frequency on Yelp 2013 data set. The review frequency value f represents the frequencies in the range $[f, f + 10)$, except when $f = 100$, where it represents the frequencies in the range $[f, \inf)$.

Models	Accuracy
NSC	51.1
BiLSTM+CSAA	52.7
HCSC	**53.8**
BiLSTM+encoder-basis-cust	50.4
BiLSTM+linear-basis-cust	50.8
BiLSTM+bias-basis-cust	51.9
BiLSTM+word-basis-cust	51.9
BiLSTM+attention-basis-cust	**53.1**

Table 6: Performance comparison of competing models in the Yelp 2013 Sparse80 data set.

a combination of CNN and the BiLSTM encoder with customization on CSAA.

Results are reported in Table 6, which provide us two observations. First, the BiLSTM model customized on the linear transformation matrix, which performs the best on the original Yelp 2013 data set (see Table 3), obtains a very sharp decrease in performance. We posit that this is because basis customization is not able to handle zero-shot cold-start entities, which are amplified in the Yelp 2013 Sparse80 data set. We leave extensions of basis for zero-shot or cold-start, studied actively in machine learning (Wang et al., 2019) and recommendation domains (Sun et al., 2012), respectively. Inspired by CSAA (Amplayo et al., 2018a), using similar review texts for inferring the cold-start user (or product), we expect to infer meta context, similarly based on similar meta context, which may mitigate the zero-shot cold-start problem. Second, despite having no zero-shot learning capabilities, Basis-Customized BiLSTM on the attention mechanism performs competitively with HCSC and performs better than BiLSTM+CSAA, which is Customized BiLSTM on attention mechanism with cold-start awareness.

6 Conclusion

We presented a new study on customized text classification, a task where we are given, aside from the text, its categorical metadata information, to predict the label of the text, customized by the categories available. The issue at hand is that these categorical metadata information are hardly understandable and thus difficult to use by neural machines. This, therefore, makes neural-based models hard to train and optimize to find a proper categorical metadata representation. This issue is very critical, in such a way that a simple concatenation of these categorical information provides better performance than existing popular neural-based methods. We propose solving this problem by using basis vectors to customize parts of a classification model such as the attention mechanism and the weight matrices in the hidden layers. Our results show that customizing the weights using the basis vectors boosts the performance of a basic BiLSTM model, and also effectively outperforms the simple yet robust concatenation methods. We share the code and data sets used in our experiments here: `https://github.com/zizi1532/BasisCustomize`.

Acknowledgments

This work was supported by Microsoft Research Asia and IITP/MSIT research grant (no. 2017-0-01779).

References

Reinald Kim Amplayo, Jihyeok Kim, Sua Sung, and Seung-won Hwang. 2018a. Cold-start aware user and product attention for sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2535–2544. Association for Computational Linguistics.

Reinald Kim Amplayo, Kyungjae Lee, Jinyoung Yeo, and Seung-won Hwang. 2018b. Translations as additional contexts for sentence classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 3955–3961.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR'15.

Andrea Baruzzo, Antonina Dattolo, Nirmala Pudota, and Carlo Tasso. 2009. A general framework for personalized text classification and annotation. In *Proceedings of the Workshop on Adaptation and Personalization for Web 2.0, AP WEB 2.0@UMAP*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659. Association for Computational Linguistics.

Po-Hao Chou, Richard Tzong-Han Tsai, and Jane Yung-jen Hsu. 2017. Context-aware sentiment propagation using LDA topic modeling on chinese conceptnet. *Soft Computing*, 21(11): 2911–2921.

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 521–526. Association for Computational Linguistics.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

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

Arash Joorabchi and Abdulhussain E. Mahdi. 2011. An unsupervised approach to automatic classification of scientific literature utilizing bibliographic metadata. *Journal of Information Science*, 37(5):499–514.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. 2008. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, ICUIMC 2008*, pages 208–211.

Fei-Fei Li, Robert Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907. Association for Computational Linguistics.

Yunfei Long, Mingyu Ma, Qin Lu, Rong Xiang, and Chu-Ren Huang. 2018. Dual memory network model for biased product review classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 140–148. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, and Xu Sun. 2017. Cascading multiway attentions for document-level sentiment classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 634–643. Asian Federation of Natural Language Processing.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Zizi Papacharissi. 2002. The presentation of self in virtual life: Characteristics of personal home pages. *Journalism & Mass Communication Quarterly*, 79(3):643–660.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. In *Proceedings of the 4th International Conference on Learning Representations*, ICLR'16.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*, pages 487–494.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Dong-ting Sun, Tao He, and Fu-hai Zhang. 2012. Survey of cold-start problem in collaborative filtering recommender system. *Computer and Modernization*, 5:59–63.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023. Association for Computational Linguistics.

Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. 2019. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):13:1–13:37.

Pengcheng Yang, Xu SUN, Wei Li, and Shuming Ma. 2018. Automatic academic paper rating based on modularized hierarchical convolutional neural network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 496–502. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016.

Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Rui Zhao, Kezhi Mao, Rui Zhao, and Kezhi Mao. 2017. Topic-aware deep compositional models for sentence classification. *IEEE/ACM Transactions on Audio, Speech and Languange Processing*, 25(2):248–260.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495.

Pengcheng Zhu and Yujiu Yang. 2017. Parallel multi-feature attention on neural sentiment classification. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pages 181–188.