Detecting Customer Complaint Escalation with Recurrent Neural Networks and Manually-Engineered Features

Wei Yang,[1,2] **Luchen Tan,**[2] **Chunwei Lu,**[2] **Anqi Cui,**[2] **Han Li,**[3] **Xi Chen,**[3] **Kun Xiong,**[2] **Muzi Wang,**[5] **Ming Li,**[1,2] **Jian Pei,**[3,4] and **Jimmy Lin**[1,2]

[1]University of Waterloo [2]RSVP.ai [3]JD.com Inc.
[4]Simon Fraser University [5]Tsinghua University

Abstract

Consumers dissatisfied with the normal dispute resolution process provided by an e-commerce company's customer service agents have the option of escalating their complaints by filing grievances with a government authority. This paper tackles the challenge of monitoring ongoing text chat dialogues to identify cases where the customer expresses such an intent, providing triage and prioritization for a separate pool of specialized agents specially trained to handle more complex situations. We describe a hybrid model that tackles this challenge by integrating recurrent neural networks with manually-engineered features. Experiments show that both components are complementary and contribute to overall recall, outperforming competitive baselines. A trial online deployment of our model demonstrates its business value in improving customer service.

1 Introduction

Customers today demand a high-quality online shopping experience, which includes prompt redress of their complaints if they are dissatisfied with any aspect of their purchase or feel their rights have been violated. Addressing such complaints is critical to building brand loyalty and preserving a company's online reputation. In most cases, complaints are first expressed to a company's customer service agents. If their dispute resolution efforts are not satisfactory, customers may seek to further escalate the complaint *beyond* the company's representatives: two common escalation scenarios are to publicly complain about their experiences on social media or to file a formal grievance with a government authority such as a consumer protection bureau. Both are obviously detrimental to an e-commerce company.

In this work, we aim to identify cases of the latter, where the customer remains dissatisfied after initial dispute resolution attempts and intends to file a formal grievance with a governmental agency. This is formulated as an online classification problem over text chat dialogues, where our goal is to preempt the escalation of the complaint by connecting the customer with a specialized service agent to intervene and provide a higher level of attention.

Our work makes the following contributions: To our knowledge, we are the first to formalize and examine this problem of identifying complaint escalation. This problem is more challenging than just performing sentiment analysis: plenty of unhappy customers express negative affect in their dialogues without escalating and filing additional grievances. Tackling this challenge requires identifying the *intent* of the customer. Our explorations began with Hierarchical Attention Networks (Yang et al., 2016), which we have adapted and simplified for our task. As with most real-world systems, our final model integrates manually-engineered features, and we show that such explicit features complement latent representations learned by recurrent neural networks. We evaluated our models both on retrospective data and in a trial online deployment. Controlled ablation studies show the contributions of neural as well as non-neural signals, and confirm that our model outperforms competitive baselines. For the academic audience, we discuss factors that impact production deployment: one important message is that despite the effectiveness of neural networks in addressing many NLP tasks, production systems often still depend on hybrid approaches that integrate manual feature engineering.

2 Background and Model Framework

The context of our problem is chat-based customer service for JD.com, a major Chinese e-commerce company. We focus on text chat initially for a

A: 您好, 京东客服1234号很高兴为您服务!
(Hi, agent id 1234 from JD.com, happy to assist you!)

C: 我再三和你们说了地址, 结果你们还让我跑去原来的地址, 这要耽误我多少天
(I've told you the address several times already, but you've wasted my time by making me go back to the original address.)

A: 您好, 还麻烦您提供下订单号, 妹子这边给您查询哦~
(Hi, can you please provide me with the order number? I'll look into this for you!)

C: 有很多单都这样, 问题一直都在. 我们消费者还能不能维权了.
(The same problem has happened on many orders. I want to protect my rights as a consumer.)

...

Figure 1: Sample chat dialogue between a service agent (A) and a customer (C).

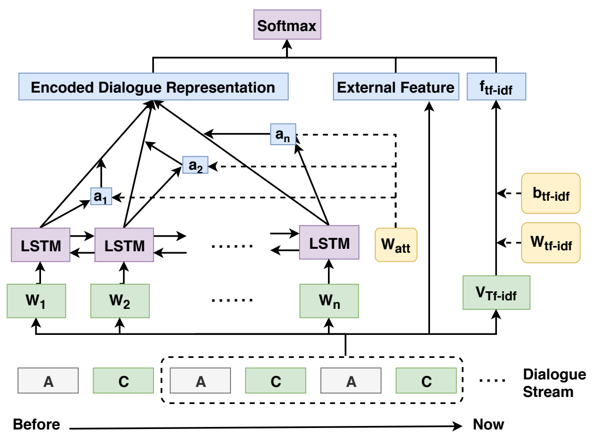

Figure 2: Our full hybrid model for detecting complaint escalation intents.

few reasons: Many customers, especially younger ones, prefer text-based interactions as opposed to speaking with customer service agents. Text chat avoids confounding errors due to imperfect speech recognition and provides an easier starting point to tackle this challenge.

We aim to deploy an automated monitoring system that continuously scans all ongoing chat dialogues in (near) real time to identify customers who intend to escalate their complaints. Separately, the company has an additional pool of specialized service agents trained to handle these more complex cases—these can be viewed as a finite resource where the monitoring system provides triage, prioritizing the attention of these agents. An angry customer, for example, might be contacted separately in an attempt to address his or her complaint to preempt the filing of additional grievances. Detecting complaint escalation intents can be viewed as a prediction problem over dialogue sequences, and our task can be modeled in terms of maximizing recall at a fixed cutoff, where the cutoff corresponds to available resources (e.g., the number of calls that these specialized service agents can make in a day).

As a result of this setup, the input to our model is a moving window of the most recent dialogue between the customer and the service agent. A sample is shown in Figure 1, along with English translations. An obvious starting point is the Hierarchical Attention Network (HAN) of Yang et al. (2016), originally developed for document classification. The model uses two separate layers of RNNs with attention mechanisms to encode context at the word- and sentence-levels, on top of which a softmax is applied for classification. This architecture was designed to capture signals at the sentence level (with the word encoder) and then integrate evidence across sentences (with the sentence encoder) to model document structure in a hierarchical manner.

Starting with the HAN architecture, preliminary experiments revealed an interesting observation—for our task, hierarchical modeling did not increase classification accuracy, as compared to an alternative, single-layer architecture. In other words, we found that a "flat" architecture that takes as input the concatenation of recent dialogue (with a special end-of-utterance token) performed just as well as HAN.

We explain this finding as follows: the sentence-level encoder attempts to learn from the sequence of utterances that comprise the chat history, which does not help in our case since most of the useful signal is concentrated at the end of the dialogue sequence. Thus, the hierarchical structure introduces more parameters without bringing much benefit. In a production environment, simpler models are preferable to more complex alternatives given comparable accuracy, and thus as a result, we use this single-layered variant of HAN as our base model. This is shown on the left side of Figure 2. However, in a slight tweak from the original HAN design, we use a bidirectional LSTM instead of a bidirectional GRU as the word encoder. We dub our model "Flat" Attention Network, or FAN for short.

The base FAN model is then augmented with several sources of additional signal from

No.	Feature	Mean	λ
1	# of emojis	0.0188	10
2	# of ellipsis marks	0.0001	10
3	# of question marks	0.1348	10
4	# of exclamation marks	0.0152	10
5	# of sentences	0.5577	10
6	# of words	0.3664	100
7	Sentiment score	0.3486	-
8	# of words in TD1	0.0051	10
9	# of words in TD2	0.0834	100

Table 1: Statistics of external textual features used in our complete model. TD1 and TD2 are two term dictionaries.

manually-engineered features. The integration of neural networks with external features in hybrid systems is common in real-world production settings for a few reasons: neural networks are often introduced to improve upon existing solutions, and hence it makes sense to reuse existing components. Manually-engineered features capture aspects of the domain that usually provide an “easy” boost in terms of accuracy. In total, we use nine features, described in Table 1.

Features 1 through 6 are self-explanatory and represent counts of various token types and simple statistics. Feature 7 is the sentiment score from a logistic regression classifier that we have separately trained on social media data. The training data contains 6.981 million Chinese microblog messages (Weibo) with at least one emoji or emoticon. The emojis and emoticons are used as (noisy) sentiment labels, e.g., happy face for positive and sad face for negative. This distant supervision method is widely used in social media (Pak and Paroubek, 2010; Lin and Kolcz, 2012). In our task, all emojis and emoticons are removed from the text during training as they only serve as the labels.

Features 8 and 9 are counts from two term dictionaries, called TD1 and TD2. TD1 contains 121 terms and was manually gathered by examining customer dialogues. TD2 contains $8,712$ terms and was extracted by computing the pointwise KL-divergence (Tan et al., 2016) between the term distributions of positive vs. negative training examples, and then selecting the top words according to this measure. TD1 is a subset of TD2.

All features (except for Feature 7) are normalized by $\min(1, f_{\text{raw}}/\lambda)$, with λ shown in the final column of Table 1. The mean values of the normalized features are shown in the third column.

In addition, we compute the tf-idf representation of the chat dialogue using only the terms from TD2 as features. From this, we wish to learn an attentive weight for each dimension (i.e., terms in TD2) and a bias parameter as follows:

$$f_{\text{tf-idf}} = W_{\text{tf-idf}} \cdot V_{\text{tf-idf}} + b_{\text{tf-idf}}$$

Finally, the nine external features in Table 1 and $f_{\text{tf-idf}}$ are concatenated to the encoded sentence representation from the base FAN model that is fed into the fully-connected layer and softmax for classification (shown in the right portion of Figure 2). The parameters $W_{\text{tf-idf}}$ and $b_{\text{tf-idf}}$ for $f_{\text{tf-idf}}$ are trained via backpropagation along with the rest of the network.

3 Experimental Setup

3.1 Dataset

For training, we gathered from the enterprise data warehouse chat logs and records of escalated complaints from February 1st to July 10th, 2018. There are approximately three million chat logs (one per customer) per day. For some complaints, we have no record of chat dialogues with the customer; these are cases where, for example, all interactions occurred over the phone. These cases comprise approximately 30% of the complaints, which we removed. This yielded a total of 21k complaints that serve as positive training examples. Of these, 45% were filed within a day after the last contact with customer service, 84% were filed within a week, and 96% within a month. From these numbers, we can see that detecting complaint escalation intents is a very hard problem, since the number of complaints is very small compared to the total volume of customer interactions; we are trying to detect a very rare event.

3.2 Model Training

Applying FAN to our problem requires making a few more design choices, since we are dealing with a dialogue between two parties: Do we take as input only the customer's text, the service agent's text, or both? Preliminary experiments show that, predictably, considering only the agent's text yields low accuracy—but somewhat surprisingly, little is gained from taking *both* the agent's text and the customer's text. We believe that this is because the agent's text contains fewer signals (since they are usually following a script), and given a fixed window size as input, it is better to maximize the amount of text from the customer that the model considers.

For positive training instances we selected the last 100 words from the customer dialogue for each complaint, under the assumption that the last interaction with the customer is the source of the complaint escalation. The window size was determined based on preliminary exploration, and we apply Jieba[1] for word segmentation. In total, we have 21k positive examples. For negative training instances, we randomly sampled dialogue data from customers who did not file a complaint. As we show in our experiments, negative sampling has an impact on the quality of our results.

We pretrained a 300-dimensional embedding for all models using fastText (Bojanowski et al., 2017) on the dialogue data in the training set with ten million negative samples. To regularize the network, we applied a dropout of 0.1 on the normalized attention weights. We used sigmoid cross entropy loss and the Adam optimizer (Kingma and Ba, 2014) to train our model. The dimension of the hidden layer in the final fully-connected network was set to 100. We reserved 10% training data for validation and found that our model reached the best recall on the validation data after three epochs of training, at which point we stopped training.

Our models were evaluated in two different ways: first, retrospectively with dialogue data extracted from the enterprise data warehouse, and second, from an online deployment.

3.3 Baselines

We compared variants of our neural model against a number of baselines:

Logistic Regression (LR): We deployed two variants, one where all tokens (about 40 million) serve as features (called LR-all) and the other where only tokens from our dictionaries (TD1 and TD2) are used (called LR-dict). In both cases, the feature vectors are weighted using tf-idf, and the model was learned using scikit learn[2] with default hyperparameters.

LightGBM:[3] We also tried a tree-based algorithm, using exactly the feature vectors as the LR-dict setting. We set the number of leaves to 32 and the maximum depth to 8. Learning rate is set to 0.2 and the number of iterations is set to 100.

Finally, we evaluated three previous neural network models, using as input the final 100 words from the customer dialogue (the same as our FAN model): fastText (Joulin et al., 2017), CNN (Kim, 2014), and LSTM (Lai et al., 2015).

3.4 Evaluation Metrics

Our classification models are evaluated as follows: We begin by gathering all the complaints filed on a particular evaluation date. The size of this set is denoted as $|C|$, which forms the ground truth for computing recall. Recall at "T-0" is computed by running our model on each customer's chat data on the day the complaint was filed at a particular cutoff K, which we refer to as R_0. Similarly, we run our model on chat data from the day before ("T-1"), two days before ("T-2"), etc. What we call "Total" recall, or R_{tot}, is computed over the union of all these $7 \cdot K$ predictions. The timespan of a week for measuring recall balances the complexity of the calculations with our observation that 84% of escalated complaints are filed within a week from last contact with the customer (See Section 3.1).

Note that our approach of selecting a particular day and looking "backwards" in time for evaluation may seem a bit counter-intuitive. A slightly more natural alternative would be to consider customer dialogues on a particular day and look "forward" in time to see if a complaint has been filed within a week. This, however, does not allow us to accurately compute the ground truth (i.e., the denominator for the recall calculation), because the customer dialogue from "today" might not be the source of the complaint. For example, the customer and the agent might have had a friendly interaction "today", and it was not until "tomorrow" that the customer became dissatisfied with some aspect of the service.

4 Results

4.1 Overall Model Effectiveness

Our main results are shown in Table 2 for data from July 17th, where each row shows the effectiveness of a model. For this evaluation, we ran our model on the last 100 words of customer dialogue at the end of the day, also extracted from the enterprise data warehouse. Here, we measure recall at $K = 3000$. The number of complaints filed on that day, or $|C|$, was 169. As we have discussed before, detecting escalation intents is a

[1] https://github.com/fxsjy/jieba
[2] https://scikit-learn.org/
[3] https://github.com/Microsoft/LightGBM

Models	R_0	R_{tot}
LR-dict	25 (14.8%)	49 (29.0%)
LR-all	22 (13.0%)	44 (26.4%)
LightGBM	22 (13.0%)	41 (24.3%)
fastText	12 (7.1%)	24 (14.2%)
CNN	27 (16.0%)	44 (26.0%)
LSTM	25 (14.8%)	52 (30.8%)
FAN$_{\text{base}}$	28 (16.6%)	60 (35.5%)
FAN$_{\text{tf-idf}}$	36 (21.3%)	69 (40.8%)
FAN$_{\text{full}}$	41 (24.3%)	75 (44.4%)

Table 2: Comparisons with baselines on July 17th, where $|C| = 169$.

difficult problem because the events are quite rare. The absolute recall numbers are difficult to interpret when attempting to answer the basic question, "Is the classifier good?" The answer to this question, however, becomes very clear when we compare FAN to the other baselines.

We evaluated three separate variants of our model: FAN$_{\text{base}}$ contains only the recurrent neural network component, FAN$_{\text{tf-idf}}$ adds the single $f_{\text{tf-idf}}$ feature (which entails learning the weights $W_{\text{tf-idf}}$ for terms in the TD2 term dictionary), and FAN$_{\text{full}}$ denotes the complete model (learning $W_{\text{tf-idf}}$ as well as taking advantage of the nine external features). In all model variants, we used 5M negative examples. The FAN base model alone beats all the other models, both those that use neural networks and those that do not. Consistent with the literature on text classification prior to the advent of neural networks, logistic regression is a simple yet strong baseline, especially coupled with feature selection: we see that using terms from the TD2 dictionary as the feature space is better than using all terms.

Looking at the recall of the neural network models, we see that fastText alone does not perform very well, and a generic CNN achieves comparable recall to logistic regression. The biggest difference between the LSTM and the FAN$_{\text{base}}$ model is incorporation of attention, and so these results show, consistent with the literature, that attention is very important. Beyond FAN$_{\text{base}}$, we see that other aspects of our model also contribute to its effectiveness. Learning the weights $W_{\text{tf-idf}}$ for terms in our TD2 dictionary alongside the recurrent neural network in an end-to-end fashion boosts recall, and manually-engineered features provide yet another boost on top of that.

# Neg	R_0	R_{tot}
10M	43 (25.4%)	73 (43.2%)
5M	42 (24.9%)	73 (43.2%)
1M	37 (21.9%)	72 (42.6%)
0.1M	31 (18.3%)	67 (39.6%)

Table 3: Effects of negative sampling on July 17th data.

Date	R_0	R_{tot}	$\|C\|$
Jul 17th	41 (24.3%)	75 (44.4%)	169
Jul 18th	30 (17.1%)	57 (32.5%)	175
Jul 19th	43 (23.1%)	87 (46.7%)	186
Jul 20st	34 (24.1%)	65 (46.1%)	141
Jul 21st	37 (38.1%)	46 (47.4%)	97
Jul 22nd	33 (24.6%)	58 (43.3%)	134
Jul 23rd	32 (20.3%)	67 (42.4%)	158
Average	36 (23.6%)	65 (43.0%)	151

Table 4: Recall results over an entire week.

4.2 Effects of Negative Sampling

Table 3 shows the impact of negative sampling on recall: Here, we fixed the positive samples and all hyperparameters, and varied the number of negative training examples. Results show that increasing the number of negative examples from 100k to 5M has a noticeable impact on recall; however, there appears to be little gained beyond 5M negative examples. From a practical perspective, the amount of negative sampling determines the training time of the model in a roughly linear manner. For this particular task, it appears that 5M represents a "sweet spot" that balances model quality and training time.

Note that the results of using 5M negative samples in Table 3 are from a different trial than the figures reported in Table 2, thus explaining small differences in results for models trained with the same settings.

4.3 Online Deployment

In Table 4, we report the recall of our model over an entire week (evaluated retrospectively). Other than outliers on July 18th and July 21st, we are able to identify approximately a quarter of all escalated complaints on that day (i.e., R_0), and the recall seems relatively stable.

In moving towards online deployment, we ran a simulation study on the July 17th data, where our classifier was run every 20 minutes on *all* ongoing customer chat dialogues. At each time step, the

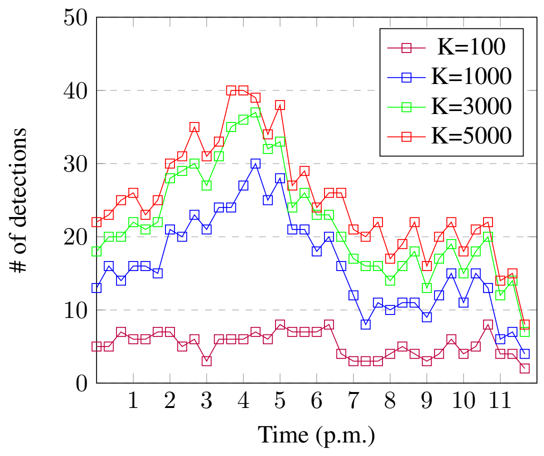

Figure 3: Simulated deployment on July 17th data.

Date	R_0	$\|C\|$
Oct 8th	52 (21.76%)	239
Oct 9th	59 (21.00%)	281
Oct 10th	54 (25.59%)	211
Oct 11th	63 (25.93%)	243
Oct 12th	50 (28.41%)	176
Oct 13th	30 (25.00%)	120
Oct 14th	43 (27.22%)	158
Average	50 (24.50%)	204

Table 5: Results from the online deployment.

classifier returns $K \in [100, 1000, 3000, 5000]$ results. Simulation output is presented in Figure 3 for the afternoon to evening hours, where the y axis shows the number of *detected* escalated complaints at that time, using all complaints filed over the next week as the ground truth. Note that as we explained above, it is not possible to compute recall because a complaint filed (for example) three days later may be based on a customer interaction that has not happened yet.

Two more caveats are necessary to properly interpret these results: First, classification output at different time steps include duplicates if a dialogue persists over a long period of time, and second, the number of successfully detected cases follows general shopping trends (e.g., late afternoon is when customer service is most active anyway). Although these results are somewhat difficult to interpret, it most closely matches our deployment scenario, since the output of our classifier at regular intervals would feed a priority queue for further enhanced customer service (and this queue would obviously remove duplicates).

Satisfied with the effectiveness of our model, it was deployed on live data starting in October 2018. We present results from a week's worth of data in Table 5. It should be emphasized that these are "real" in-the-wild results from actual online dialogues (whereas all previous results were on retrospective data extracted from the data warehouse). These result are compiled by running our classifier every 20 minutes with $K = 5000$, but R_0 is computed with respect to the union of all results after deduplication (so these values are not directly comparable to previous tables). Also, note that while R_{tot} is useful as a retrospective metric, it does not make sense to measure in an online setting. In terms of R_0 as a business metric, we estimate that we have improved recall by 12–15% over the previous model, based on comparable reports in July.

5 Related Work and Discussion

Detection of complaint escalation intents is straightforwardly formulated as a text classification problem, which of course has been studied for decades. Prior to the neural wave, popular techniques include Naive Bayes (McCallum and Nigam, 1998) and Support Vector Machines (Joachims, 1998) with feature engineering. There is plethora of work based on CNNs (LeCun et al., 1998; Kim, 2014; Johnson and Zhang, 2015; Conneau et al., 2017) and RNNs (Johnson and Zhang, 2016; Zhou et al., 2015; Socher et al., 2013); attention mechanisms have also been found to be effective (Yang et al., 2016; Du et al., 2017; Du and Huang, 2018).

We readily concede that there are at best minor modeling advances in this work and thus little novelty from a purely academic perspective. However, our primary contribution is to provide a case study to the broader community of how NLP solutions are deployed in production settings. In this respect, we make two points:

First, we feel that many models discussed in the academic literature are too complex for operational deployment: model complexity increases training time, inference latency, and sensitivity to hyperparameters, which can make models unstable, particularly to incoming data that is changing and evolving. This is why we are constantly trying to simplify models without compromising quality—for example, this led to our observation that for our task, hierarchical modeling (Yang et al., 2016) does not seem to contribute tangible value. In production deployments, there are important tradeoffs between complexity, accuracy, and inference latency that need to be considered, and we do not see much discussion along these lines in the academic literature. As an example of the last consideration, taking the last N words of customer dialogue represents a compromise, since our model needs to monitor *all* ongoing dialogues at a particular moment in time (numbering in the tens of thousands).

Second, our deployed model is a mishmash of manually-engineered features, external dictionaries, and multiple neural components. It certainly lacks the "elegance" of end-to-end neural solutions that dominate the literature, but we dare say that most deployed "real world" systems are complex hybrids like ours. Most important business problems are not solved *de novo*: there are usually already-deployed solutions we are trying to improve upon, in which case it makes no sense to ignore existing features and models and start from scratch. In the academic literature, hybrid approaches are under-explored relative to their real-world impact. It would be desirable to see more papers that examine the evolution of approaches from, for example, rule-based systems to manual feature engineering to neural models.

6 Conclusion

There is often a chasm between research and practice, and this is certainly the case for many NLP applications. Through this work, we hope to build a bridge between academic researchers and industrial practitioners by sharing some of our experiences in designing and deploying hybrid models combining neural networks and feature engineering. For the specific problem of detecting customer complaint escalation intents, we have shown that while neural networks have indeed advanced the state of the art, manual feature engineering still contributes to effectiveness and still has its place in the "toolbox" of the practitioner.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116.

Changshun Du and Lei Huang. 2018. Text classification research with attention-based recurrent neural networks. *International Journal of Computers Communications & Control*, 13(1):50–61.

Jiachen Du, Lin Gui, Ruifeng Xu, and Yulan He. 2017. A convolutional attention model for text classification. In *Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing*, pages 183–195.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using LSTM for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 526–534.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: a method for stochastic optimization. *arXiv:1412.6980*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Jimmy Lin and Alek Kolcz. 2012. Large-scale machine learning at Twitter. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 793–804.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*, pages 41–48.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010)*, pages 1320–1326.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Luchen Tan, Adam Roegiest, Charles L.A. Clarke, and Jimmy Lin. 2016. Simple dynamic emission strategies for microblog filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1009–1012.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A C-LSTM neural network for text classification. *arXiv:1511.08630*.