IITP at IJCNLP-2017 Task 4: Auto Analysis of Customer Feedback using CNN and GRU Network

Deepak Gupta*, **Pabitra Lenka**†, **Harsimran Bedi***, **Asif Ekbal***, **Pushpak Bhattacharyya***
*Indian Institute of Technology Patna, India
†International Institute of Information Technology Bhubaneswar, India
*{deepak.pcs16, harsimran.mtcs16, asif, pb}@iitp.ac.in
†pabitra.lenka18@gmail.com

Abstract

Analyzing customer feedback is the best way to channelize the data into new marketing strategies that benefit entrepreneurs as well as customers. Therefore an automated system which can analyze the customer behavior is in great demand. Users may write feedbacks in any language, and hence mining appropriate information often becomes intractable. Especially in a traditional feature-based supervised model, it is difficult to build a generic system as one has to understand the concerned language for finding the relevant features. In order to overcome this, we propose deep Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) based approaches that do not require handcrafting of features. We evaluate these techniques for analyzing customer feedback sentences on four languages, namely English, French, Japanese and Spanish. Our empirical analysis shows that our models perform well in all the four languages on the setups of IJCNLP Shared Task on *Customer Feedback Analysis*. Our model achieved the second rank in French, with an accuracy of 71.75% and third ranks for all the other languages.

1 Introduction

Exploration and exploitation of customer feedbacks have become highly relevant and crucial for all the customer-centric business firms in today's world. Product manufacturers would like to know what their customers are liking or complaining about in order to improve their services or launch improved versions of products. Service providers would like to know how happy or unhappy a customer is with their service. According to a survey[1] 96% of unhappy customers do not complain but they advise 15% of their friends to not have any business dealings with the particular firm. With the huge amount of feedback data available, it is impossible to manually analyze each and every review. So there arises a need to automate this entire process to aid the business firms in customer feedback management.

A customer review analysis can be associated with its sentiment polarity ('positive', 'negative', 'neutral' and 'conflict') or with its interpretation ('request', 'comment', 'complaint'). There exists a significant number of works for sentiment classification (Pang et al., 2002; Glorot et al., 2011; Socher et al., 2013; Gupta et al., 2015; Deepak Gupta and Bhattacharyya, 2016), emotion classification (Yang et al., 2007; Li and Lu, 2009; Padgett and Cottrell, 1997) and customer review analysis (Yang and Fang, 2004; Mudambi and Schuff, 2010; Hu and Liu, 2004). However, the meaning of customer reviews (*request, complaint, comment* etc.) remains a relatively not much explored area of research.

Our present work deals with classifiyng a customer review into one of the six predefined categories. This can be trated as a document classification problem.

The classes are *comment, request, bug, complaint, meaningless* and *undetermined*. The feedback classification is performed across four different languages, namely English (**EN**), French (**FR**), Japanese (**JP**) and Spanish (**ES**). In Table 1 we depict few instances of customer feedbacks with their label(s) in different languages. One of critical issues in traditional supervised model is to come up with a good set of features that could be ef-

[1] https://goo.gl/8KVwBh

Proceedings of the 8th International Joint Conference on Natural Language Processing, Shared Tasks, pages 184–193,
Taipei, Taiwan, November 27 – December 1, 2017.

fective in solving the problem. Hence it is challenging to build a generic model that could perform reasonably well acoross different domains and languages. In recent times, the emergence of deep learning methods have inspired researchers to develop solutions that do not require careful feature engineering. Deep Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are two very popular deep learning techniques that have been successfully used in solving many sentence and document classification (Kim, 2014; Xiao and Cho, 2016) problems. We aim at developing a generic model that can be used across different languages and platforms for customer feedback analysis.

The remainder of our paper is structured as follows: Section 2 offers the related literature survey for customer feedback analysis, where we discuss about the existing approaches. Section 3 describes our two proposed approaches, one based on CNN and the other based on amalgamation of RNN with CNN. Section 4 provides the detailed information about the data set used in the experiment and the experimental setup. Results, analysis and discussion are elucidated in Section 5. We put forward the future work and conclude the paper with Section 6. The source code of our system can be found here.[2]

2 Related Work

Analysis of customer feedback has been of significant interest to many companies over the years. Given the large amount of feedbacks available, interesting trends and opinions among the customers can be investigated for many purposes. In this section, we discuss the related literature in the analysis of customer feedbacks.

Bentley and Batra (2016) developed a Office customer voice (OCV) system that classifies customer feedback on Microsoft Office products into known issues. The classification algorithm is built on logistic regression classifiers of the Python scikit framework. They have also employed a custom clustering algorithm along with topic modeling to identify new issues from the feedback data. A domain specific approach described in (Potharaju et al., 2013) infers problems, activities and actions from network trouble tickets. The authors developed a domain specific knowledge base and an ontology model using pattern mining and statistical Natural Language Processing (NLP) and used it for the inference. Brun and Hagège (2013) offers a pattern to extract suggestions for improvements from user reviews. They combine linguistic knowledge with an opinion mining system to extract the suggestive expressions from a review. The presence of a suggestion indicates that the user is not completely satisfied with the product.

Over the years, many companies have developed feedback management systems to help other companies gain useful insight from the customer feedback data. Customer satisfaction survey done by Freshdesk [3] defines metrics for measuring customer satisfaction. They use a five class categorization ('positive', 'neutral', 'negative', 'answered' and 'unanswered'), thereby combining sentiment and responsiveness. Survey Monkey[4] also facilitates us to create survey forms and analyze them for customer satisfaction. It also uses a five-class categorization ('excellent', 'good', 'average', 'fair' and 'poor'), a commonly used rating mechanism. Customer Complaints Management[5] provides services and softwares to help business firms to manage and retain their customers.
Identifying a category for customer feedback requires deep semantic analysis of the lexicons to identify the emotions expressed. Authors in (Asher et al., 2009) have provided detailed annotation guidelines for opinion expressions where each opinion lies in the four top level categories of *'Reporting', 'Judgement', 'Advise' and 'Sentiment'*.
With the advent of deep learning, in recent years there has been a phenomenal growth in the use of neural network models for text analysis.Yin et al. (2016) have provided practical and effective comprehensive relevance solutions in Yahoo search engine. They have designed ranking functions, semantic matching features and query rewriting techniques for base relevance. Their network model incorporates a deep neural network and it is tested with the commercial Yahoo search engine.

In this shared task the organizers have introduced a customer feedback analysis model where the task is to classify a feedback into one of the six categories. In following section we describe our proposed deep learning based classification model

[2] https://github.com/pabitralenka/Customer-Feedback-Analysis

[3] https://freshdesk.com/

[4] https://www.surveymonkey.com/

[5] http://www.newgensoft.com/solutions/cross-industry-solutions/customer-complaints-management/

Feedback	Class(es)
nouveau bug : le rayon led anti yeux rouges se declanche intempestivement après avoir envoyé la photo !	bug
あと、タイムラインで他の人がお気に入りの記事を表示しないように設定できるようにしたい	request
Saw advertisements through an Internet travel booking site for hotel and zoo tickets in a package deal.	comment
La decoración en el hotel es excelente y las habitaciones tienen un toque chic.	comment
it is fast, but the controls are lousy, plus it keeps installing on my desktop shortcuts to place I don't want.	complaint
Mi pareja y yo hicimos una escapada romántica a Barcelona de cuatro días.	meaningless
Pour moi et avec modesties d'éloges, nero multimedia me rassure en ce sens que je peux:	undetermined
編集で付けようとしても、どうしてもその人のだけ消えてしまう	bug, comment

Table 1: Some instances of customer feedback in different languages annotated with their class(es)

for customer feedback analysis.

3 Network Architecture for Feedback Classification

In this section we describe our proposed neural network architecture for feedback classification. We propose two variants, the first one is convolution operation inspired CNN and the second one is the amalgamation of CNN with RNN.

3.1 Feedback classification using CNN

In this model a feedback sentence is subjected to CNN and the model predicts the most probable feedback class along with the confidence (probability) score. The model architecture is depicted in Fig 1. The input and output of the model are as follows:

INPUT: A feedback F, labeled with any of the six classes: (*comment, request, bug, complaint, meaningless,* and *undetermined*)

OUTPUT: Class(es) of the corresponding F

Our model uses similar network architectures used by Kim (2014) for performing the feedback classification task. We depict our model architecture in Figure 1. The architecture of a typical CNN is composed of a stack of distinct layers where each layer performs a specific function of transforming its input into a useful representation. A CNN comprises of *sentence representation*, one or more *convolutional layers* often interweaved with *pooling layer* (max-pooling being extremely popular), followed by *fully-connected layer* leading into a $softmax$ classifier. The components of CNN are described as follows:

3.1.1 Feedback Sentence Representation

As CNNs deal with fixed length inputs, we ensure that every input feedback sentence has the same length. To achieve this, input feedback sentences are padded according to the need. Each feedback sentence is padded to the maximum sentence length[6]. Padding of feedback sentences to the same length is useful because it allows us to efficiently batch our data while training. Let a feedback sentence *F* consisting of 'n' words be the input to our model such that $x = [x_1, x_2, \ldots x_n]$ where x_i is the i^{th} word in the feedback sentence. Each token $x_i \in F$ is represented by its distributed representation $p_i \in R^{\text{k}}$ which is the k-dimensional word vector. The distributed representation p is looked up into the word embedding matrix W which is initialized either by a random process or by some pre-trained word embeddings like *Word2Vec* (Mikolov et al., 2013) or *GloVe* (Pennington et al., 2014). We then concatenate (row wise) the distributed representation p_i for every i^{th} token in the feedback F and build the feedback sentence representation matrix. The feedback sentence representation matrix $p_{1:n}$ can be represented as:

$$p_{1:n} = p_1 \oplus p_2 \oplus \cdots \oplus p_n \quad (1)$$

where $\oplus$ is the concatenation operator. Each row of the sentence representation matrix corresponds to the word vector representation of each token. The result of the embedding operation yields a 3-dimensional tensor.

3.1.2 Convolutional Layer

The convolutional layer is the core building block of a CNN. The common patterns (n-grams) in the training data are extracted by applying the convolution operation. These patterns are then passed to the next hidden layer to extract more complex patterns, or directly fed to a standard classifier (usually a $softmax$ layer) to output the final prediction. The convolution operation is performed on the feedback representation matrix *via* linear fil-

[6]maximum feedback sentence length: EN=116, FR=73, JP=7 and ES=92

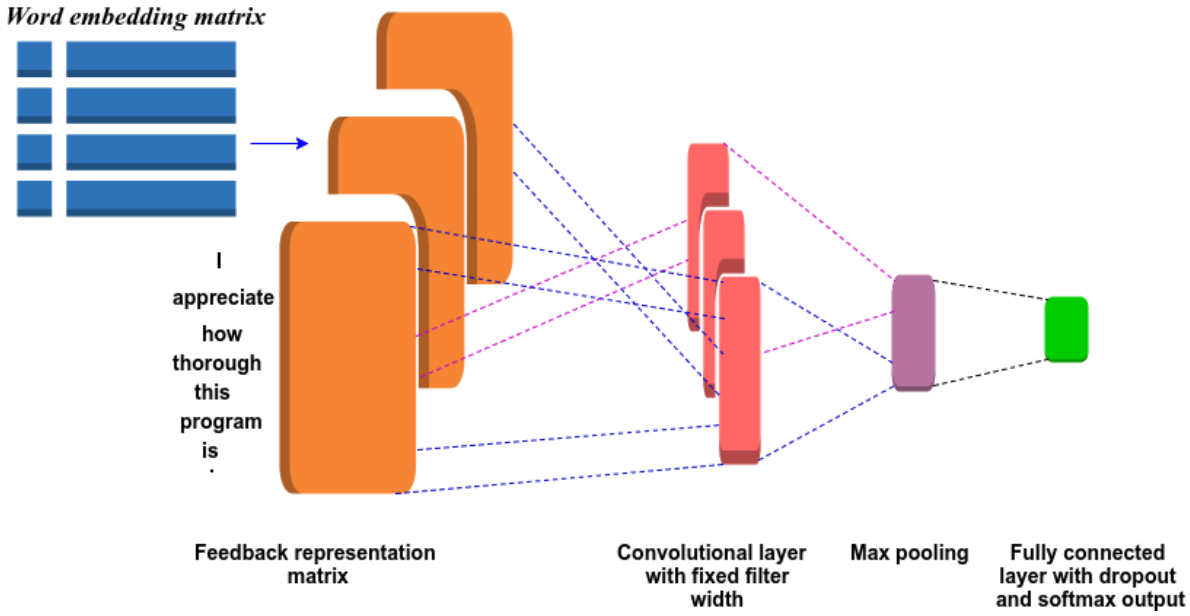

Figure 1: Convolution neural network based feedback classification model

ters (feature detectors). Owing to the inherent sequential structure of the text data, we use filters with fixed *width*. Then we simply vary the *height* of the filter, i.e. the number of adjacent rows (tokens) considered together. Here the *height* of the filter is the region size of the filter.

We consider a filter parameterized by the weight matrix w with a region size h. Thereafter we denote the feedback representation matrix by $S \in R^{n \times k}$, where k is the dimension of the word vector. The generated output $out_i \in R^{n-h+1}$ of the convolutional operator is obtained by repeatedly applying the filter on sub-matrices of S:

$$out_i = w \cdot S[i : i + h - 1], \quad (2)$$

The sub-matrix of S from i^{th} row to $i + h - 1^{\text{th}}$ row is represented by $S[i : i + h - 1]$, where $i = 1 \ldots n - h + 1$ and h is the height of the filter. A bias term $b \in R$ and an activation function f to each out_i is added which generates the *feature map* $c \in R^{n-h+1}$ for this filter where:

$$c_i = f(out_i + b) \quad (3)$$

But the dimensions of the *feature map* produced by each filter will differ as a function of the number of words in the feedback sentence and the filter region size. Thus we apply a pooling function over each *feature map* to generate a vector of the fixed length.

3.1.3 Pooling Layer

The output of the convolutional layer is the input to the pooling layer. The primary utility of the pooling layer lies in progressively reducing the spatial dimensions of the intermediate representations. The operation performed by this layer is also called *down-sampling*, as there is a loss of information due to the reduction of dimensions. However, such a loss is beneficial for the network for two reasons:

1. Decreases the computational overhead of the network; and
2. Controls over-fitting.

The pooling layer takes a sliding window or a certain region that is moved in stride across the input which transforms the values into representative values. There are several pooling operations in practice such as *max pooling, min pooling, average pooling* and *dynamic pooling*. We have applied the *max pooling* operation (Collobert et al., 2011) on the feature map which transforms the representation by taking the maximum value from the values observable in the window. Max pooling has been favored over others due to its better performance. It also provides a form of translation invariance and robustness to position. However, Springenberg et al. (2014) have proposed to discard the pooling layer in the CNN architecture.

3.1.4 Fully-connected Layer

Fully-connected layers are typically used in the final stages of the CNN to connect to the output layer. It looks at what high level features most strongly correlate to a particular class. The features generated from the pooling layer p form the penultimate layer and are fed to a fully connected softmax layer to generate the classification. The $softmax$ classifier gives an intuitive output (normalized class probabilities) and also has a probabilistic interpretation. The output of the $softmax$ function is the probability distribution over tags (*comment, request, bug, complaint, meaningless,* and *undetermined*).

$$P(c = i|F, p, z) = softmax_i(p^{\mathrm{T}} w_i + z_i) = \frac{e^{p^T w_i + z_i}}{\sum_{k=1}^{K} e^{p^T w_k + z_k}} \quad (4)$$

where z_k and w_k are the bias and weight vector of the k^{th} labels.

3.2 Feedback Classification using CNN coupled with RNN

We propose a second method for feedback classification that combines both CNN and RNN. The typical architecture of these combinations is composed of a *convolutional feature extractor* applied on the input, then a *recurrent network* on top of the CNN's output, then an optional *fully connected layer* is added to RNN's output and finally fed into the *softmax layer*. We use convolutional layer as it learns to extract higher-level features that are invariant to local translation. By stacking up multiple convolutional layers, the network can extract higher-level, abstract, (locally) translation invariant features from the input sequence.

Apart from this advantage, it is noticed that several layers of convolution are required to capture long-term dependencies, due to the locality of the convolution and pooling. In order to capture long-term dependencies even when there is only a single layer present in the network, the recurrent layer comes handy. However, the recurrent layer increases the computational overhead due to its linearly growing computational complexity with respect to the length of the input sequence. So we have used a combination of convolutional and recurrent layers in a single model to ensure that it can capture long-term dependencies from the input more efficiently for the feedback classification task. Our model is similar to the one proposed in Xiao and Cho (2016). They have used LSTM unit, however we have employed GRU (Cho et al., 2014).

$$\mathbf{z}_i = \sigma(\mathbf{W}_z c_i + \mathbf{V}_z \mathbf{h}_{i-1} + \mathbf{b}_z)$$
$$\mathbf{r}_i = \sigma(\mathbf{W}_r c_i + \mathbf{V}_r \mathbf{h}_{i-1} + \mathbf{b}_r)$$
$$\mathbf{c}_i = tanh(\mathbf{W} c_i + \mathbf{V}(\mathbf{r}_i \odot \mathbf{h}_{i-1}) + \mathbf{b})$$
$$\mathbf{h_i} = z_i \odot \mathbf{h}_{i-1} + (1 - \mathbf{z}_i) \odot \mathbf{c}_i$$

where $\mathbf{z}_i$, $\mathbf{r}_i$ and $\mathbf{c}_i$ are update gate, reset get and new memory content, respectively. c_i is the convolution output at time t. We take the last hidden states of both directions and concatenate them to form a fixed-dimensional vector, which are later fed into the next layer.

4 Datasets and Experimental Setup

4.1 Datasets

The data sets used in our experiments are provided by the organizers of the shared task on *Customer Feedback Analysis* of IJCNLP-2017. Data sets consist of representative real world samples of customer feedback from Microsoft Office customers in four languages, namely *English, French, Japanese* and *Spanish*. We obtain the English translations of test data of other three languages (*French, Japanese* and *Spanish*) which were translated using Google translate[7]. Each feedback in the data is annotated with one or multiple tags from the set of six tags (*comment, request, bug, complaint, meaningless* and *undetermined*). We show the dataset statistics for each language in Table 2.

4.2 Data Preprocessing

Some of the feedback sentences are annotated with multiple classes. Before feeding them to the network, we preprocessed the data by replicating the particular instance with all the possible classes.

4.3 Regularization

In order to prevent the model from over-fitting, we employed a dropout regularization (set to 50%) proposed by Srivastava et al. (2014) on the penultimate layer of the network. It "drops out" a random set of activations in the network. Dropout prevents feature co-adaptation by randomly setting some portion of hidden units to zero during the forward propagation when passing it to the softmax output layer in the end to perform classification. It also

[7] https://translate.google.com/

Language	Training							Development							Test						
	CO	CP	RQ	BG	ME	UD	Total	CO	CP	RQ	BG	ME	UD	Total	CO	CP	RQ	BG	ME	UD	Total
EN	1758	950	103	72	306	22	**3211**	276	146	19	20	48	3	**512**	285	145	13	10	62	4	**519**
ES	1003	536	69	14	9	0	**1631**	244	39	12	5	1	0	**301**	229	53	14	2	1	0	**299**
FR	1236	529	38	53	178	10	**2044**	256	112	6	8	36	1	**419**	255	104	11	8	40	2	**420**
JP	826	531	97	89	0	45	**1588**	142	73	22	18	0	9	**264**	170	94	26	14	0	9	**313**
Total	**4823**	**2546**	**307**	**228**	**493**	**77**	**8474**	**918**	**370**	**59**	**51**	**85**	**13**	**1496**	**939**	**396**	**64**	**34**	**103**	**15**	**1551**

Table 2: Data set statistics of all the languages. Notations used are defined as follows, **CO:** *comment*, **CP:** *complaint*, **RQ:** *request*, **BG:** *bug*, **ME:** *meaningless* and **UD:** *undetermined*,

forces the network to be redundant i.e. it should be able to provide the correct classification or output for a specific input even if some of the activations are dropped out.

4.4 Network Training and Hyper-parameters

We have applied the rectified linear units (ReLu) (Nair and Hinton, 2010) as the activation function in our experiment. We use the development data to fine-tune the hyper-parameters. In order to train the network, the stochastic gradient descent (SGD) over mini-batch is used and Backpropagation algorithm (Hecht-Nielsen, 1992) is used to compute the gradients in each learning iteration. We have not enforced L2 norm constraints on the weight vectors as Zhang and Wallace (2015) found that the constraints had a minimal effect on the end result. We have used cross-entropy loss as the loss function. The hyper-parameters of the best system in each language are listed in Table 6.

4.5 Experiments

We conduct experiments in two different ways: CNN based and CNN+RNN based. Further we perform the experiments with *original test data* and *English translated test data*. In each setting we experiment with all the four languages[8]. Through the experimental results we wanted to establish the fact that whether a simple machine translation would work or there is a need of native tools for the other languages. The following are the descriptions of our submission in the shared task.

1. **CNN:** The results obtained from the CNN model described in Section 3.1.
   (a) **With original test data:** We train the CNN model using the dataset provided for training and used the respective test data to obtain the results. This setting of experiments are employed on all four languages (EN, FR, JP and ES).
   (b) **English translated test data:** We train the CNN model using the English training dataset and use the English translated test data of other languages (FR, JP and ES) to obtain the results.
2. **CNN+RNN:** The results obtained from the CNN+RNN model described in Section 3.2.
   (a) **With original test data:** Similar to CNN we train the CNN+RNN model using the dataset provided for training and use the respective test data to obtain the results. This setting of experiments are employed on all the four languages (EN, FR, JP and ES).
   (b) **English translated test data:** Again similar to CNN, we train the CNN+RNN model using the English training dataset and use the English translated test data of other languages (FR, JP and ES) for the evaluation.

We use the pre-trained Google word embedding[9] to initialize the word embedding matrix for English. The word embedding matrix for other three languages are initialized randomly[10].

5 Results and Discussions

We have submitted our results using both the models as discussed in Section 4.5. Table 3 summarizes the performance of both the models *with original test data*. Table 4 summarizes the performance of both the models on *English translated test data*. Table 7 shows the exact accuracy comparison with the models which achieve the best accuracy as compared to our model and also with the

[8] In the "English translated test data" setting, the experiments were performed with three languages (FR: French, JP: Japanese and ES: Spanish)

[9] https://code.google.com/archive/p/word2vec/

[10] due to some computational issues, we were unable to use pre-trained embeddings of other languages.

Language	Tags	CNN			CNN + RNN		
		Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Japanese	comment	0.564	0.965	0.711	0.569	0.994	0.724
Japanese	complaint	0.167	0.011	0.020	0.333	0.011	0.021
French	comment	0.789	0.867	0.826	0.785	0.886	0.832
French	complaint	0.630	0.558	0.592	0.603	0.452	0.516
French	request	-1	0	-1	1	0.091	0.167
French	meaningless	0.577	0.375	0.455	0.531	0.425	0.472
Spanish	comment	0.917	0.913	0.915	0.906	0.930	0.918
Spanish	complaint	0.597	0.755	0.667	0.656	0.755	0.702
Spanish	request	1	0.286	0.444	1	0.214	0.353
English	comment	0.826	0.818	0.822	0.713	0.775	0.743
English	complaint	0.611	0.738	0.669	0.538	0.593	0.564
English	request	1	0.077	0.143	0.625	0.385	0.476
English	meaningless	0.667	0.452	0.538	0.500	0.177	0.262

Table 3: Performance results of the model *with original test data* at the tags level. We did not provide the results of those tags which were not detected by our model.

Language	Tags	CNN			CNN + RNN		
		Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Japanese	comment	0.808	0.741	0.773	0.706	0.776	0.739
Japanese	complaint	0.571	0.766	0.655	0.552	0.511	0.530
Japanese	request	-1	0	-1	0.667	0.154	0.250
Japanese	bug	-1	0	-1	0.500	0.071	0.125
French	comment	0.837	0.863	0.849	0.766	0.875	0.817
French	complaint	0.679	0.692	0.686	0.651	0.538	0.589
French	request	1.000	0.091	0.167	-1	0	-1
French	meaningless	0.433	0.325	0.371	0.435	0.250	0.317
Spanish	comment	0.915	0.895	0.905	0.901	0.913	0.907
Spanish	complaint	0.636	0.792	0.706	0.603	0.660	0.631
Spanish	request	-1	0	-1	0.500	0.071	0.125

Table 4: Performance results of the model with *English translated test data* at the tags level. We did not provide the results of those tags which were not detected by our model.

baseline scores. Our systems easily predicted the true labels of sentences which had either positive connotation words like "great", "pleasant", "nice", "good", etc or negative connotation words like "not", "slow", "unable", "horrible", etc and classified them into *comment* and *complaint* classes respectively. The negative connotation words also appeared in the feedback sentences of *bug* class. But owing to the larger amount of training data in the *complaint* class as compared to the *bug* class, the negative connotation words appeared significantly in the *complaint* class. As a result, our systems had difficulty in predicting the true labels for the feedback sentences associated with the *bug* class. Our systems were unable to detect some tags due to the class imbalance problem in the training as well as test data. The scores of our systems could have been much better, provided that we should have more labeled training data. The system performance can be improved by the language specific pre-trained word embeddings.

5.1 Error Analysis

We perform error analysis on the outputs of our best performing model. Our system failed to detect some of the true positive classes due to some inadequacy in the training data. Table 5 provides some examples (from different languages) where our system fails to detect the correct tags. We divide those inadequacy into three different cate-

Error Type	Language	Feedback	Reference	Predicted
Ambiguous	EN	Make a paid version so we don't have to deal with the ads	request	complaint
Ambiguous	ES	La verdad, ir, ir, no va mal.	comment	complaint
Ambiguous	FR	Une bouilloire et du thé et du café dans la chAmbiguousre (ainsi que sucre et lait). La salle de bain était grande.	comment	complaint
Ambiguous	JP	その後大浴場でサンセットを見てゆっくり入浴	comment	complaint
Missing Target Entity	EN	Work with any type of PDF	comment	meaningless
Missing Target Entity	ES	El agua salpicaba el suelo del baño.	complaint	comment
Missing Target Entity	FR	de la grosse arnaque !	complaint	meaningless
Missing Target Entity	JP	英語の勉強にもなりそう	comment	meaningless
Too short	EN	I gave up.	comment	complaint
Too short	ES	Decepcionante	complaint	meaningless
Too short	FR	Brunch en famille	meaningless	comment
Too short	JP	使えん	complaint	meaningless

Table 5: Some of the feedback instances from different languages where our model failed to predict the correct tags.

Parameter Name	EN	ES	FR	JP
Embeddings	Pre-trained	Random	Pre-trained	Pre-trained
Maximum epochs	100	200	100	100
Mini batch size	64	64	64	64
Number of filters	128	128	128	128
Filter window sizes	3,4,5	3,4,5	3,4,5	3,4,5
Dimensionality of word embedding	300	300	300	300
Dropout keep probability	0.5	0.5	0.5	0.5
Hidden unit size (CNN+RNN)	-	300	-	-

Table 6: Network hyper-parameters for the best system (ref: Table 7) in each language

	Accuracy		
Language	Best System	Our Best System	Best Baseline
EN	71.00%	70.00% (CNN)	48.80%
ES	88.63%	85.62% (CNN+RNN)	77.26%
FR	73.75%	71.75% (CNN-Trans)	54.75%
JP	75.00%	63.00% (CNN-Trans)	56.67%

Table 7: Performance comparison with the best system in the shared task and the best baselines (3-gram features based SVM classifier). **CNN-Trans**: CNN model with English translated test data

gories:

- **Ambiguous Feedback**: Ambiguous feedback sentences have several possible meanings or interpretations. Our system fails to comprehend such doubtful or uncertain nature of customer feedback.

- **Missing Target Entity in Feedback**: We found some feedback which were pretty straight without having a particular subject entitled to it. These type of feedback sentences fail to address about what is being referred to in the sentences. These sentences do not sound complete. Let's take an example : "Work with any type of PDF". It does not specify any comprehensive meaning. So the questions like "What will work?", "What is being talked about?" are bound to come up when the feedback sentences have an unstated subject. This, in turn, generates misclassification.

- **Too Short Feedback**: There are several shorter-length feedback sentences, which are generic and do not provide any good evidence. Sometimes, these type of feedback sentences fail to convey any proper meaning to the end user who deals with it. The systems also experience difficulty to correctly tag those feedback sentences.

6 Future Work and Conclusion

Convolutional neural networks (CNN) and recurrent neural networks (RNN) are architecturally two different ways of processing dimensioned and ordered data. These model the way the human visual cortex works, and has been shown to work incredibly well for natural language modeling and a number of other tasks. In our work, we extensively made use of CNN and RNN (GRU) to

perform classification of customer feedback sentences into six different categories. Our proposed model performed well for all the languages. We have performed thorough error analysis to understand where our system fails. We believe that the performance can be improved by employing pre-trained word embeddings of the individual languages. Future work would focus on investigating appropriate deep learning method for classifying the short feedback sentences.

References

Nicholas Asher, Farah Benamara, and Yvette Yannick Mathieu. 2009. Appraisal of opinion expressions in discourse. *Lingvisticæ Investigationes*, 32(2):279–292.

Michael Bentley and Soumya Batra. 2016. Giving voice to office customers: Best practices in how office handles verbatim text feedback. *2016 IEEE International Conference on Big Data (Big Data)*, pages 3826–3832.

Caroline Brun and Caroline Hagège. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70:199–209.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Asif Ekbal Deepak Gupta, Ankit Lamba and Pushpak Bhattacharyya. 2016. Opinion mining in a code-mixed environment: A case study with government portals. In *International Conference on Natural Language Processing*, pages 249–258. NLP Association of India.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Deepak Kumar Gupta, Kandula Srikanth Reddy, Asif Ekbal, et al. 2015. Pso-asent: Feature selection using particle swarm optimization for aspect based sentiment analysis. In *International Conference on Applications of Natural Language to Information Systems*, pages 220–233. Springer, Cham.

Robert Hecht-Nielsen. 1992. Neural networks for perception (vol. 2). chapter Theory of the Backpropagation Neural Network, pages 65–93. Harcourt Brace & Co., Orlando, FL, USA.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

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, Doha, Qatar. Association for Computational Linguistics.

Mu Li and Bao-Liang Lu. 2009. Emotion classification based on gamma-band eeg. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 1223–1226. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Susan M Mudambi and David Schuff. 2010. What makes a helpful review? a study of customer reviews on amazon. com.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Curtis Padgett and Garrison W Cottrell. 1997. Representing face images for emotion classification. In *Advances in neural information processing systems*, pages 894–900.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rahul Potharaju, Navendu Jain, and Cristina Nita-Rotaru. 2013. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 127–142, Berkeley, CA, USA. USENIX Association.

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.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR*, abs/1602.00367.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 275–278. IEEE.

Zhilin Yang and Xiang Fang. 2004. Online service quality dimensions and their relationships with satisfaction: A content analysis of customer reviews of securities brokerage services. *International Journal of Service Industry Management*, 15(3):302–326.

Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 323–332, New York, NY, USA. ACM.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820.