MLP Coursework 4: Word-level and Character-level Representation for Sentiment Analysis

G46: s1700808, s1718204, s1780366

Abstract

In this project, we extend baseline models on word-level to LSTM, GRU and the combination of CNN and LSTM. Then we investigate different character-level representations(one-hot and index) on different network structures (CNN and LSTM). Finally, word and character models are combined to see if combined word-and-char-level features can be used to provide additional information to improve the performance of both single word-level and char-level models. By conducting these experiments, we found that: 1) the combination of CNN and LSTM models gain good performances on different settings. 2) the char-level implementation we adopt needs large scale of dataset thus it has poor performance on the small IMDB data. 3) the char-level features are beneficial to the word-level models and it can provide extra information to improve the performance.

1. Introduction

Machine learning has dramatically advanced the performance of sentiment analysis tasks. Since the performance of machine learning method is heavily dependent on the choice of text representations (Bengio et al., 2013), much of the effort in sentiment analysis goes into designing the feature representation of text. Most of the effort has focused on the word level. However, word representation has out-of-vocabulary(OOV) problem (Jean et al., 2014).Meanwhile, in sentiment analysis, some different spelling variants like 'looong' may contain emotional information, which are usually pre-processed in word-level, such as ignoring the expression or reverting it to the original normalization form. Many previous works focused on fixing this issue by using character-level representation and are proved to be efficient on a large number of NLP tasks (Golub & He, 2016; Santos & Zadrozny, 2014; Chung et al., 2016). The success of character-level representation motivated us to explore whether character-level representation could perform better on sentiment analysis by solving the problem of OOV and the way dealing with emotional expressions.

We put forward **two research questions** to explore. **1)** Explore different word representations on different classifiers in IMDB dataset (Maas et al., 2011). **2)** Investigate whether character-level model can perform better on sentiment analysis by mitigating the OOV problem on word-level and providing way to deal with emotional expressions like elongated words in the same dataset.

In order to explore above research questions, we have **four objectives**: **1)** Investigate different word representations. We will construct different word representation models including pre-trained models on IMDB dataset(i.e. CBOW, Skip-gram) and Wikipedia documents (i.e. GloVe) with simple classifiers in IMDB dataset as our baselines. In addition, we explore on whether the domain variation of corpus would affect the quality of embeddings. **2)** Extend our baseline model to recurrent nets in order to investigate whether this state-of-the-art could boost the classification performance. **3)** Compare the behavior of character-level model on IMDB test data with our baseline models and fancier word level models to see if character-level representation has better performance. **4)** Combine word-level and character-level model to see whether combined models could outperform either sole word-level or character-level model.

In coursework 3, the first objective is done. We conducted experiments comparing the performances of different word embeddings on our CNN model, GOOGLE $fastText$ model(Joulin et al., 2016) and other basic statistical representations on SVM models on word level. It illustrated that Google $fastText$ model achieves the best performance among all, and TFIDF representation outperforms other statistic representations (i.e. one-hot, BOW, frequent) on SVM classifier. We also found that the domain of dataset has significant influence on embedding quality because GloVe embedding that trained on Wikipedia documents (Pennington et al., 2014) performed worst while CBOW embeddings (Mikolov et al., 2013) trained on IMDB performed best. In coursework 4, we will continue exploring our research questions and objects. Comparing with the objects we put forward in coursework 3, we added the fourth object: exploring the performance of combined word-and-character-level models to see whether combined model can bring more information for sentiment classification.

Next, we will introduce the methodology (which is not covered in coursework 3) for the project. Section 3 will show the experiments we setup, then we will compare and analyze the experiments results of models with different representations on sentiment classification task. In section 4, we will link our results with other published works and give some ideas for future work. Section 5 illustrates the overall summary and discussion for the whole project.

2. Methodology

In coursework 3, we have introduced the word-level representation and classifier in details. So in this section, we will introduce the character-level representation, combined word-and-character-level models and Gated Recurrent Unit(GRU).

2.1. Character-level

Character-level representation maps the character to vectors of real numbers and we will use two methods to construct character-level representations: one-hot character representation and index character representation.

One-hot character representation This method is proposed by Zhang et al. (2015) , which is also called character quantization. We adopt a slightly different version which will be described in details later. It is done by prescribing an alphabet for the input language, and then use one-hot encoding for each character. Then, the sequence of characters is transformed to a sequence of vectors with fixed length. Character sequences within this fixed length will be maintained as the input, and any characters that are not in the curated alphabet are quantized as all-zero vectors. The alphabet we created consists of 69 characters, including 26 English letters, 10 digits, 32 other characters and the new line character. The non-space characters are:

a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9
, : ; ! ? ' " / \ | @ % # $ ˆ & * ~ ' + - = <> () []

Each character in the input sequence is represented as one-hot encoding (1*69 vector).

Index character representation This method is quite similar to the simple index word embedding as we used in coursework 3, it represents each character with the integer of its frequency ranking order. We directly represent sentences by these indexes and use them as the input value for the embedding layer, where embedding layer randomly initialized the initial vector representation with Gaussian probability function for each input. Finally, neural models will update their values over time.

2.2. Combined word-and-char-level model

The combined word-and-character-level model learns word-level and character-level representations in two separate branch, and concatenate their separate output to do prediction. We will combine the best model on world level and character level and pass their concatenated output to a dense layer to predict the sentiment polarity. It will be explained in more details in section 3.

2.3. GRU

As a variant of LSTM, GRU has gate units that modulate the flow of information inside the unit(Cho et al., 2014a). However, it does not have a separate memory cells ans only has two gates: one is the update gate and another is the reset gate (Chung et al., 2014).GRU combines the input gate and forget gate in LSTM as update gate. The update gate is used to control the state information at last moment being brought into the current state. The greater the value of the update gate, the more information will be used to change. The reset gate is to control the degree of state information that ignores the previous moment. The smaller the value of the reset gate is, the more previous information will be ignored.

3. Experiments and Evaluation

3.1. Basic setup

Based on the experiments of coursework 3, we continue to conduct experiments on word-level and character-level representations with different neural models on IMDB dataset. we use the same data preprocessing methods we used in coursework 3. In the following models using word-level representations, we keep the most frequent 20,000 words among all dataset and use the first 1,000 tokens for each sentences as the input data. We still use the CBOW and Skip-gram embeddings trained on IMDB dataset for word-level experiments.

Meanwhile, when dealing with character level representations we use two different approaches. **1) one-hot**, we adopt basically the same vocabulary as Zhang et al. (2015) proposed because they use similar datasets of online reviews but of much larger scale (like Yelp review and Amazon review). Thus we assume they share similar features and their vocabulary is also suitable for our data. The vocabulary we use is slightly different from theirs, and their vocabulary has two '-' symbols, one for minus symbol and another one for hyphen. It is hard to distinguish these two symbols and makes no difference to exclude one '-' out and get a vocabulary of 69 characters. Similarly, we also adopt their setting for input sequence length of 1014 as Zhang et al. (2015) suggested. **2) index representation**, we represent characters in terms of their rankings of occurrence frequency and combine these indexes to represent the whole sentence. The sequence length we use is still 1014 (same as said one-hot approach). However, we keep all the characters in the IMDB dataset thus get a vocabulary of 200 characters which covers all char inputs in our dataset. For this representation, our embedding layer will convert each char index into randomly initialized vector according to its distinct index. It also allows the embedding value to be updated by back-propagation.

More detailed experimental settings are described in following specific experiment sections. The overall experiment results on validation set and test set are shown as table 2.

3.2. Baseline models

In coursework 3, we run BOW, one-hot, frequency and TF-IDF on SVM classifier, the skip-gram, CBOW and GloVe on the CNN classifier with dropout rate of 0.5, kernel size of 3, dense units of 250 and CNN filter of 100. We also run the most recent method ($fastText$ with bi-gram) for word representation. We found that the $fastText$ model achieves the best performance among all. In addition, TFIDF representation outperforms other statistical representations with SVM and even GLOVE_CNN. CBOW_CNN performs better than other models except $fastText$ model with bi-gram features. We choose TFIDF_SVM, CBOW_CNN and FastText_bigram as our baselines. These three models are evaluated on test set and the results are shown in Table 2.

3.3. Word-level LSTMs

Motivation CNNs are good at extracting spatial features, whilst in NLP field, the word order is very important which is something CNNs cannot handle. However, RNNs can encode the sequence inputs and take the word ordering into account while traditional RNNs are proved to be difficult in training back-propagation through time (BPTT) (Rumelhart et al., 1986). Simple RNNs cannot handle the long-term dependency within sentences because of vanishing gradient problem, which will make the sentiment analysis perform worse in some cases. Theoretically, LSTMs can deal with this problem and capture the information of token orders by harnessing three different gates: input gate, forget gate and output gate. With these gates, LSTMs are capable of capturing the sequential information of natural language sentences and making decisions about the data flow between LSTM states (Olah, 2015). Hence, we investigate the effect of LSTM structures with word-level representations using previous trained CBOW and Skip-gram embeddings.
Description We trained our experiments on LSTMs of 1 layer with no dropout, no regularization and binary cross entropy loss function. Different models with LSTM units of 32, 64, 128 respectively are training and compared according to the loss and accuracy on dev set. The optimal model is selected according to the accuracy on validation set. After choosing the LSTM model hyper-parameters according to the performance on dev set, we also test and report the performance of optimal LSTM model with word-level representation on test set.
Results The results (see figure 1 for the performance on training set and dev-set) indicate that LSTMs with 128 hidden units outperform others on our IMDB dataset, gaining a higher validation accuracy and remaining stable trend of validation accuracy after epoch 3. In addition, LSTMs with CBOW embeddings have a better performance than models with Skip-gram embedddings. Hence, we choose LSTMs with **1 layer and 128 units** on **CBOW embeddings** as the optimal settings.
Interpretation and discussion As table 2, the optimal model of LSTM with word-level embedding achieves 87.91% accuracy on test set, which is 2.47% worse than the counterpart of previous CNN1D model. The first reason might be the small size of our dataset which are not able to utilize the advantage of LSTM. In our experiments, the input data has been padded or cut to have the same length, where some sequential dependencies may be lost. In addition, sentiments of movie reviews are often determined by certain part of the phrases or words in review sentences, where the locality could be easily captured by CNN model (Yin et al., 2017). As mentioned by Dauphin et al. (2016), CNN, as a hierarchical model, might outperform RNNs when the input is finite context information rather than unbounded context.

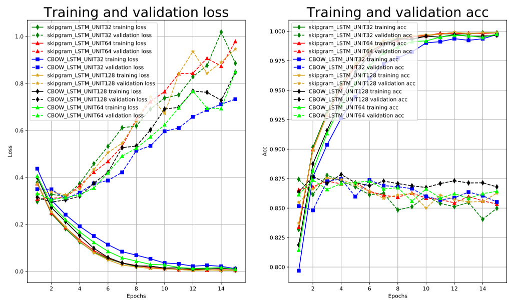

Figure 1. Experiment results of LSTM with word-level representation.

3.4. Word-level GRUs

Motivation As a variant of LSTMs, GRUs simplify the computation of LSTMs by merging the input and forget gates together as well as the cell states and hidden states (Cho et al., 2014b). We also replace LSTM structures with GRUs in order to see whether there are changes on the performance of sentiment classification task.
Description Like previous LSTM settings, we maintain the hyper-parameters of LSTMs: 1 layer GRU plus 1 dense layer, no dropout and penalty with GRU unit size of 32, 64 and 128. We conduct our experiments with CBOW and Skipgram embeddings and plot the changing trend of loss and accuracy on training set and dev set (see figure 2 for the GRU performance).
Results As the left graph in figure 2, GRUs using CBOW embeddings outperform those using Skip-gram embeddings. Moreover, GRUs with 64 unit size using CBOW embeddings (as the red lines in figure 2) behave better than others on validation loss after epoch 2 and illustrate to have a comparably stable trend w.r.t. validation accuracy. Hence, we chose GRUs CBOW with 64 hidden states as the optimal setting for word-level GRU experiments.
Interpretation and discussion It can be prove that the performance of GRU model on word-level is similar to that of LSTM model (as table 2), where GRU model gain 0.18% lower on validation set whilst get 0.16% higher on test set comparing with LSTM model in terms of the accuracy. It is also consistent with our expectation that LSTMs and GRUs are well-matched and can achieve similar results.

3.5. Word-level CNN+LSTM & LSTM+CNN

Motivation Since CNNs could acquire spatial information but cannot capture the sequential ordering information as LSTMs, we combined the previous two experiments together by adding the optimal LSTM layers before and after the optimal CNN-1D layer. Firstly, we tried CNN-LSTM model (see figure 3 for schematic illustration) where CNNs adopt our pretrained CBOW embedding as input, reduce the

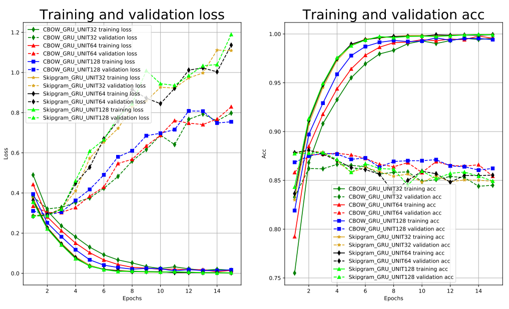

Figure 2. Experiment results of GRU with word-level representation.

dimension with max-pooling to capture spatial information, followed by a LSTM layer to encode the token sequence context. In addition, by reversing the order of LSTM layer and CNN layers, LSTM-CNN model applies LSTM to encode the CBOW embeddings at first, then harnesses CNNs to learn the local features. It is expected to perform better than sole CNNs or LSTMs.

Description Like said in word-level experiments, we still use the optimal CBOW embedding on our combined models, where CNN layers has 1 layer with filter number 100, kernel_size 3, followed by a maxpooling layer with step size 4, and then LSTM has 128 units. There is always a dense layer with unit 1 and sigmoid activation function (rather than softmax because of 2 categories of sentiments) after either CNN+LSTM or LSTM+CNN structure.

Results As can be seen in table 2, the CNN+LSTM models reached 91.27% accuracy on test set, surpass previous CNN1D, LSTM and GRU models with CBOW embeddings by 0.89%, 3.36% and 3.2% respectively. Meanwhile, LSTM_CNN model behave 0.1% better than CNN_LSTM model on the accuracy on test set, which also out perform previous models.

Interpretation and discussion In this experiments, the results of combined CNN and LSTM model illustrated that out initial hypothesis is correct. As for CNN_LSTM model, the front CNN layer could extract local features and pool it into smaller dimensions, after which LSTM layer could encode the sequence ordering. This model could learn more on our movie review sentences. Similarly, in the LSTM_CNN model, LSTM layer could be regarded as the encoder of CBOW embeddings by encoding the sequential ordering information and then CNNs are used to extract the local features which may determine whether the sentiment is positive or negative. To sum up, both said combined models outperform previous single NN models as we expected.

3.6. Character-level CNN: one-hot/index

Motivation As indicated in the introduction section, the word-level embedding methods have drawbacks when tackling OOV problem, and it also needs special process on occasions like elongated words which are commonly seen informal words in social media. From many recent papers and experiments, the character-level representation has gained good improvement on different NLP tasks

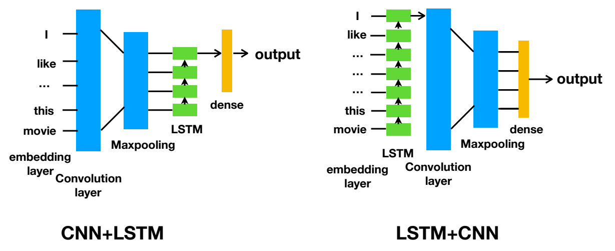

Figure 3. Schematic diagram of combined-CNN-and-LSTM models (The left is CNN+LSTM model while the right is LSTM+CNN model).

compared to word-level models (Zhang et al., 2015). Our IMDB dataset consists of movies reviews from the social media, so the problems mentioned above are more severe on our sentiment classification task. Thus, it is worth trying character-level representation models to seek improvements.

Description Following Zhang et al.'s experiments, we reproduce the same structure of the character-level CNN model. It consists of 6 CNN layers each with a maxpooling layer and 3 fully connected layers with dropout in between. We first use the one-hot character-level representation to conduct experiments and then also try the index representation after finding the optimal network structure for our data. The hyper-parameters we use for this model are presented in table 1, which is the same to the Zhang et al.. Two dropout layers are inserted into 3 dense layers with 0.5 dropout rate. This structure is the original one from the paper and the paper uses a much larger dataset which might make the model too complex for our data. But we just use it for a starter, and then to reduce the size to make it more suitable for our small scaled dataset. As the adjustments of the structure are based on previous experimental results, we put the process in the result section.

Results The original structure has very bad performance of approximately 50.1% (all following accuracies are on val set unless specified) of accuracy as shown in figure 4. We do expect such bad result because our data is too small for such a complex network structure. In order to improve the performance on the network, we firstly reduce the CNN layer to 4 layers by removing layer 4 and 5 as numbered in Table 1. The result is basically the same (50.1%) with the full structure. Then we experiment with models by decreasing CNN layers to 3, 2, 1 (removing CNN layers from the second to end, alway keep the first layer). The 3CNN-layer model gains great improvement with 83.3% accuracy on val set while the 2CNN layer model goes back to 50.1% accuracy. We think this might be caused by the large number of dense layer and hidden units. So we first try to remove one dense layer (layer No.7) on 2CNN layer model to seek improvements. The DNN-layer-reduced model has much better performance than the previous one with 84.5% of accuracy which yields the best result currently. Then we try to reduce the hidden units to 128 filters for CNN and 512 units for DNN. The new model still get good performance of 83.2% but slightly worse than the previous experiment. After this, we keep reducing units to 128 filters for CNN and 256 units for DNN which yields worse result of 82.7%, so that we stop exploring the units number and keep them 256 for CNN and 1024 for DNN. Running this setting on the 1 CNN layer model, we get the result of 72.4%. We also try hyper-parameters with 128 filters for CNN, 256 units for DNN and 128 filters for CNN, 512 units for DNN on the model with 1CNN layer and 2DNN layer, but little improvement can be seen. As a result, we choose the **2CNN and 2DNN layer model (including sigmoid classification layer) with 256 filters for CNN and 1024 units for DNN** as the optimal configuration (we select the best configuration in terms of validation accuracy). After selecting the optimal structure, we change the representation to character-level index representation (described in section 3.1) and run the experiment on the optimal model. The index representation model gives result of 50.2% and the model's performance on test set is 49.93% as presented in Table 2.

CNN layers			
Layer	Feature map	Kernel	Pool Size
1	256	7	3
2	256	7	3
3	256	3	N/A
4	256	3	N/A
5	256	3	N/A
6	256	3	3
Dense Layers			
Layer	Output units	activation	
7	1024	relu	
8	1024	relu	
9	1	sigmoid	

Table 1. The full structure of the character-level CNN network

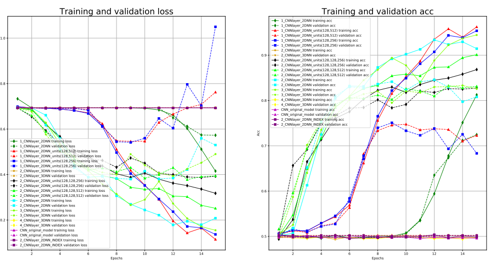

Figure 4. The results of the CNN_character_level model with different layers, units and representation.

Interpretation and discussion As described above and shown in the figure 4, the model with 2CNN layer, 2DNN layer, 258 filters for CNN and 1024 units for DNN is the optimal (84.5% accuracy on test set). As the figure shows, the complexer the CNN model is, the larger scale of dataset we need to get a decent result. Zhang et al. (2015) achieved pretty good result on different classification tasks by the original full CNN model while we only obtain an so-so result on the small-sized model which is even much worse than the previous word-level CNN model (90.09%). In that paper, they regard characters as signals and thus take advantage of the property that CNNs are very useful in extracting information from raw signals. While to make this effective and useful, large-scale datasets are required. Zhang et al. use several different datasets and the smallest one has 120,000 training samples while the largest has more than 3 million samples. Compared to that, we only have 35,000 training samples which we think is the main reason to the poor performance. In fact, Zhang et al. have already made the conclusion that the character-level ConvNets start to do better only when the dataset goes up to the scale of several millions. Our experimental results also fulfill this conclusion. As to the pretty bad performance of the index representation model, this is quite a surprising result. The index constructed sentence has less sparseness compared to the one-hot encoding, it makes us think that this might help the model perform better. While the totally different result makes us quite surprised. The bad performance we think might be caused by the vocabulary of total 200 characters employed by index representation. The index representation contains many strange and unusual characters while the one-hot encoding only chooses 69 characters that are commonly seen. This might bring confusion to the CNN model and thereby result of a worse performance. Actually, most of the time the sentiment polarity can be expressed by particular or specific phrases which are commonly seen, thus the 69-common-character vocabulary can be utilized more properly.

3.7. character level LSTMs: one-hot/index

Motivation Dai & Le (2015) claimed that LSTMs are good at encoding sequential inputs. Motivated by their LSTM experiments on character level, we further to conduct experiments applying LSTM models on our IMDB dataset. Based on the character-level representation we applied in previous char_CNN model, we continue to investigate the impact of LSTM model with both one-hot and index representation described in section 3.1.

Description The LSTM experiments on Dai & Le (2015) adopts DBpedia dataset with 2 layer LSTM layers with 512 units in each layer. Due to the fact that DBpedia dataset has 560,000 training data and 70,000 test data,approximately 10 times larger than ours (our preprocessed IMDB dataset has 35,000 training set, 7,500 validation set and 75,00 test set). Also, DBpedia has 14 categories whilst IMDB only has two classes. In addition, applying same hyper-parameters (especially 512 units for LSTM layer) make our device crash. Hence, we try to reduce the hyper-parameter to 1 layer with a range of LSTM units: 32, 64 and 128. The input data is also either one-hot matrix or index representation for chars. We did not adopt dropout and regularization, which is the same as the experiment settings of Dai & Le (2015).

Results The binary classification accuracy and loss on training and validation set are shown as figure 5. It is obvious that LSTM models with one-hot char representation performed badly. In contrast, index representation outperform one-hot representation in terms of accuracy on validation set, among which LSTMs with 128 units perform best. Hence, char LSTM model with 128 units using index representation is selected to be the optimal in this experiment (79.32% accuracy on test set).

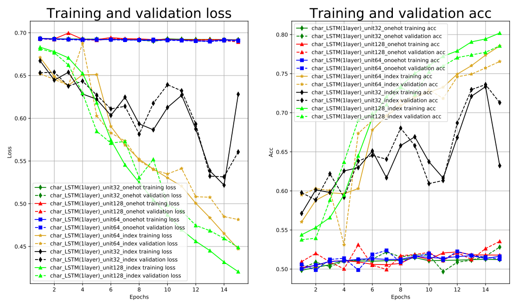

Figure 5. Experiment results of LSTM with one-hot character-level representation.

Interpretation and discussion Obviously, index representation outperform one-hot representation when applying LSTMs on char level. Firstly, utilizing one-hot matrix does not perform well seems to be because of the sparsity of input data. What's more, as Zhang et al. (2015) suggested, we just keep the first 1014 characters for each movie review paragraph. It seems that LSTM will be affected to a large extent because of the sparse data problem. Also, index representations adopts normal distribution as random initialization pdf, which could be optimized by BPTT process. In order to make LSTMs perform better, LSTM layer might need to receive pre-trained dense embeddings rather than sparse one-hot embeddings.

Dai & Le (2015) also proposed a Sequence Autoencoder (SA) LSTM model. With this SA-LSTM approach, firstly train an encoder-decoder RNN architecture with input sequence as both input and output data. Then harness the weights of encoder as the initial initialization of our LSTM layers. In other words, keep the encoder layer and add a dense layer to predict the sentiment polarity. This SA method provides our LSTM model a shortcut to optimize the gradients for recurrent nets. We have implemented it[1], but cannot smoothly run it due to the system crash of our devices. It might need more GPU resource for encoder-decoder architecture and we might continue to explore it in future work.

3.8. Character level CNN+LSTM & LSTM+CNN: one-hot/index

Motivation We have already conducted experiments on word-level CNN+LSTM and LSTM+CNN in section 3.5, and the results of these two models are pretty good, just slightly below the $fastText\_bigram$ model. Inspired by previous experiments, we assume these two models will also have good performance on character-level. So we transfer our previous models to character-level to test our assumption.

Description The character-level CNN model we use here is the same from the optimal one in section 3.6 and the char-LSTM models we use are the optimal ones selected from section 3.7 for one-hot and index representation respectively. Since we still explore the difference between index and one-hot representation, we conduct four different experiments by combining two representations and two classifiers. Thus, these four experiments are: CNN+LSTM with one-hot, CNN+LSTM with index, LSTM+CNN with one-hot, LSTM+CNN with index.

Results The results are shown in figure 6 and table 2. The Char_index_CNN+LSTM gains the best performance on test set of 87.09% accuracy. The other three models also gain good performance on test set compared to previous character-level experiments.

Interpretation and discussion From the table 2, it is noticeable that the CNN+LSTM model with either representations outperforms the LSTM+CNN model with same setting. We think the reason might be that in CNN+LSTM model, the features sent to LSTMs have smaller dimensions after applying pooling and LSTM could learn the local features extracted by CNN. In terms of LSTM+CNN models, LSTM layer might not encode the sequential information efficiently because we just keep the top 1014 chars for every sentence and this could lose some ordering information. As to the one-hot representation, all experiments with one-hot have worse performance compared to index. This might be caused by the sparseness of one-hot encoding.

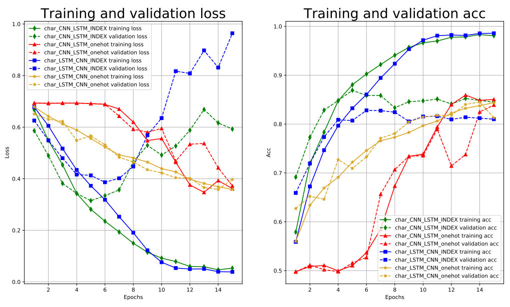

Figure 6. Experiment results of LSTM_CNN and CNN_LSTM with different character-level representation.

3.9. Combined character level & word level CNN/LSTM

Motivation After conducting above character-level experiments, even the best result on test set is only 87.09% by Char_index_CNN+LSTM model as shown in Table 2. The result is much worse than the simple word-level CNN model. The main reason we think is the small-scaled dataset, so we give up solely relying on the character-level features to obtain good performance. Alternatively, we consider using character-level features to aid the word model to get better performance. Our hypothesis is that the character-level features can indicate information of irregular words and markers, thus can help the word-level model to tackle problems like elongated words. So we

[1]see assignment code sub-folder for the SA-LSTM script: `CYK/char_enc_dec.py`

expect the result of combined model could be better.

Description For this combined experiment, we use two different kinds of classifier structures. The first one is the combined word-and-char CNN model and the second one is the combined word-and-char LSTM. Due to the complexity of the CNN+LSTM model, the runtime will be extremely long if we feed combined features into it, so we just explore the simplest CNN and LSTM. **1) CNN**, the word-level model we use is the word-CNN model (with CBOW embedding, 90.38% on test set) which is the baseline and one of the best performed model from coursework 3. And we choose the best performed char_CNN model (84.50% on test set) which is with the one-hot representation to combine with. The char_CNN model is the optimal model selected from section 3.6. For both model, we remove their dense layers and concatenate the output of these two models before feeding it to one fully-connected layer with 512 hidden units and then add dropout layer with 0.5 dropout rate. Finally we pass the results to sigmoid classification layer. The combined structure is shown in Figure 7. **2) LSTM**, the word-LSTM model we use is the optimal model selected from section 3.3 which consists of 1 LSTM layer with 128 units (CBOW embedding). The result of the model (87.91%) on test set is presented in Table 2. In order to make comparison with CNN model, the char_LSTM model we use is still the one-hot encoding model selected from section 3.7 which yield a bad result of 50.92% on test set. The model consists of 1 LSTM layer with 128 units. We first remove sigmoid layer for these two models and add it right after concatenating them.

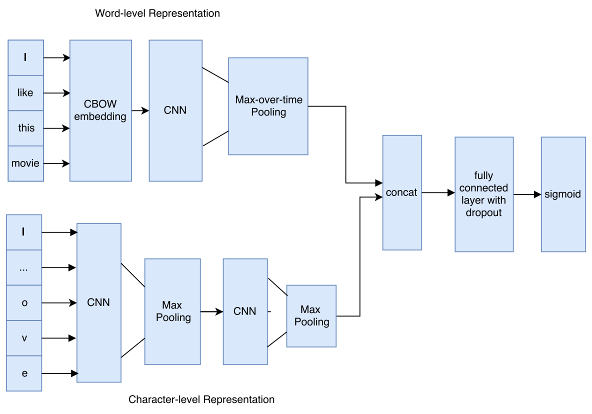

Figure 7. Architecture for combined word-and-character CNN_Model.

Results The results are shown in Table 2 (last two rows), The word_char_one_hot_CNN model achieves 90.80% of accuracy on test set, which outperforms both the single word_CNN model and the single char_1-hot_CNN model on both validation and test set. This is also true for word_char_one_hot_LSTM model, the performance of the combined LSTM model is better than that of either single model on both validation and test set.

Interpretation and discussion From the Table 2, the combined models all have better performances compared to either single model which indicates that the adding character-level features can provide essential information to word features and improve the classification accuracy. We did not specially design experiments to test OOV phenomenon and elongated words, so from above results we can only say that the character-level features might have impact on relaxing those problems, but they definitely can help improve the performance of word-level model. Interestingly, the char_1-hot_LSTM model actually has very poor performance (50.92% on test set), but the character-level features extracted by the model also improve the performance of the corresponding word-level model. It indicates that adding character-level feature might always be a good way to obtain better performance.

Word-level representation				
Model	Loss (val)	Acc (val)	Loss (test)	Acc (test)
TFIDF_SVM	-	0.8943	-	0.8969
fastText_bigram	0.2240	0.9183^{14th}	0.2207	**0.9176**
Word_CNN	0.2563	0.9009^{5th}	0.2407	0.9038
Word_LSTM	0.1995	0.8755^{3rd}	0.3156	0.8791
Word_GRU	0.3336	0.8737^{4th}	0.3182	0.8807
Word_CNN+LSTM	0.2486	0.9060^{3rd}	0.2247	0.9127
Word_LSTM+CNN	0.2422	0.9049^{2nd}	0.2270	**0.9137**
Character-level representation				
Model	Loss (val)	Acc (val)	Loss (test)	Acc (test)
Char_1-hot_CNN	0.3847	0.8457^{12th}	0.3876	0.8450
Char_index_CNN	0.6932	0.502^{1st}	0.6932	0.4993
Char_1-hot_LSTM	0.6922	0.5233^{5th}	0.6928	0.5092
Char_index_LSTM	0.4489	0.7855^{15th}	0.4372	0.7932
Char_1-hot_CNN+LSTM	0.3722	0.8390^{15th}	0.3654	0.8446
Char_index_CNN+LSTM	0.3341	0.8585^{6th}	0.3030	**0.8709**
Char_1-hot_LSTM+CNN	0.3584	0.8452^{14th}	0.4363	0.8039
Char_index_LSTM+CNN	0.3863	0.8281^{6th}	0.3741	0.8328
Combined word-and-character models				
Model	Loss (val)	Acc (val)	Loss (test)	Acc (test)
Word+char_1-hot_CNN	0.2478	0.9036^{9th}	0.2380	**0.9080**
Word+char_1-hot_LSTM	0.2931	0.8893^{6th}	0.2902	0.8873

Table 2. The optimal performance on different models. The performance with best accuracy on validation set is shown, where the superscripts on validation accuracy indicate the number of epoch that picks the values. The best accuracy on test set in each set of experiment is marked as bold font (in above table, **above dashed line** are **baseline models** in coursework 3).

4. Related Work

Sentiment analysis has been handled as a Natural Language Processing task at many text levels. From document level (Turney, 2002; Pang & Lee, 2004), paragraph level (Le & Mikolov, 2014), sentence level (Hu & Liu, 2004; Kim & Hovy, 2004) and phrase level (Wilson et al., 2005) to more frequently at word level. The paper(Pang et al., 2002) pioneered word level by representing each word as a one hot vector. However, this representation loses much of the meaning on the semantic level. Recently, a large number of works tried to find a better word representation for sentiment analysis. Bespalov et al. (2011) initialized the word embedding by Latent Semantic Analysis and further represent each document as the linear weighted of n-gram vectors for sentiment classification in Amazon and TripAdvisor. Severyn & Moschitti (2015) used an unsupervised neural language model to train initial word embeddings that are further tuned by deep learning model on a Twitter supervised corpus. However, there are some issues that word-level model may suffer, like out-of-vocabulary problem and the way handling the emotional expressions. Many previous works in other tasks fixed this by using character-level model. Santos & Zadrozny (2014) proposed a DNN that learns character-level representation for POS tagging task. Also, Zhang et al. (2015) demonstrated that character-level CNN is effective in text classification. Dai & Le (2015) also demonstrated that character-level SA-LSTM and linear gain can reduce the error rate in document classification.

For our experiments, we chose to work on IMDB dataset because movie reviews may magnify the problem of word-level since it has many emotional and informal expressions. Besides, Turney (2002) found that movie reviews are most difficult among several domains for sentiment classification. Meanwhile,Dai & Le (2015) did sentiment analysis experiments on IMDB dataset and found that the accuracy of SA-LSTM with word-level embedding model can achieve 92.76%. Le & Mikolov (2014) proposed an unsupervised algorithm called Paragraph Vector to learn fixed-length feature representations from variable-length pieces of texts and got 92.58% on IMDB dataset.

In our experiments, we constructed different word-level and character-level representations on different architecture of classifiers followed the instruction of Dai & Le (2015) and Zhang et al. (2015) to figure out whether character-level model can work better than word-level model on sentiment analysis with IMDB dataset. However, according to our results, character-level model didn't perform better than word-level model like other tasks. We then tried to combine word-level and character-level,and it proved that the combined model outperforms its original single model. There are some other works also trying to combine word-level and character-level models. Liang et al. (2017) used multi-channel GRU units and CNN architecture to learn the representations of word-level and character-level, and project it to a softmax output layer for relation classification. Nakov & Tiedemann (2012) explored several combinations of character and word-level translation models for translating between closely-related languages with scarce resources.

our works show that in the sentiment analysis task on IMDB dataset, simple character-level model can not outperform word-level model and the combination of the two level although perform better than only one level model, it still fail to exceed $fastText$. In the future, we can try some state-of-art character-level representation, like hierarchical RNN architectures consisting of multiple modules with different timescales(Liang et al., 2017).

5. Conclusion

In this project, we investigate the impact of different representations on word-level and character-level as well as different architecture of classifiers. Firstly, we used SVM with TF-IDF matrix, Google $fastText$ with bi-gram features, CNN1D with CBOW embeddings as our baseline. When choosing baseline systems, we also trained our CBOW and Skipgram embeddings on IMDB dataset, compared with GloVe trained on wikipedia data. It can be seen that domain-specific embeddings (i.e. CBOW and Skip-gram here) outperform out-of-domain embeddings (i.e. GloVe), hence we continue to do experiments using domain-specific embeddings on our word-level experiments. Then, further word-level experiments was conducted on different structures of NN, including LSTM and its variants as well as different combinations of LSTMs and CNNs. It is illustrated that combined LSTM and CNN models outperform other structures, and CBOW embeddings surpass Skipgram embeddings on word-level models. Afterwards, referred to Zhang et al. (2015), character-level one-hot vectors were used to represent our character inputs. Moreover, motivated by the very simple word-level representation, we also use random initialized char representation (namely index representation as section 3.1). It shown that one-hot representations perform better than random initialized representations on CNN model whilst w.r.t LSTMs, it is just the opposite. Finally, we hypothesize that it might help if we combined the char-level model and word-level model altogether by concatenating the final output of each model and add a dense layer to predict. The overall results can be seen as table 2, combined model of both CNNs and LSTMs could behave better, which prove that our initial intuition is reasonable. Overall, except Google $fastText$ model, the optimal accuracy is obtained by Word_LSTM+CNN model (91.37% on test set).

Our four objectives are completed: 1) We conducted experiments comparing the performances of different word embeddings on our CNN model, GOOGLE $fastText$ model and other basic statistical representations on SVM models on word level. 2) we extend baseline models on word-level to LSTM, GRU and the combination of CNN and LSTM. 3) Different character-level representations(one-hot and index) are investigated on different network structures. 4) We construct word-character combined models which outperform either single model. Since the results of character-level models don't outperform word-level models while the word-character model does have better performance than either single word or character model on IMDB dataset. These results also answer our research questions. It is proved that character-level features can help improve the performance of word-level model.

Our experiments showed that the performance of char-level is not as good as word-level on IMDB dataset. There are several possible reasons about this. Firstly, we just adopt the one-hot and random initialized character representation, which are not efficient and very state-of-art char-representations. There are still a large space for us to improve in case exploring more char-level representations. In addition, as proposed by Dai & Le (2015), sequence autoencoder could be used to gain a better initialization for recurrent models. Further more, IMDB contains only 50,000 movie reviews, which is too small compared with yelp (Asghar, 2016) and DBpedia (Auer et al., 2007). Char embeddings are proved to be efficient when training on large size of dataset (Zhang et al., 2015; Hwang & Sung, 2017). We might also use a larger dataset to train our character-level model to acquire a better performance.

References

Asghar, Nabiha. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*, 2016.

Auer, Sören, Bizer, Christian, Kobilarov, Georgi, Lehmann, Jens, Cyganiak, Richard, and Ives, Zachary. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pp. 722–735. Springer, 2007.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Bespalov, Dmitriy, Bai, Bing, Qi, Yanjun, and Shokoufandeh, Ali. Sentiment classification based on supervised latent n-gram analysis. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 375–382. ACM, 2011.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014a.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014b.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Chung, Junyoung, Cho, Kyunghyun, and Bengio, Yoshua. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*, 2016.

Dai, Andrew M and Le, Quoc V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

Dauphin, Yann N, Fan, Angela, Auli, Michael, and Grangier, David. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

Golub, David and He, Xiaodong. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*, 2016.

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

Hwang, Kyuyeon and Sung, Wonyong. Character-level language modeling with hierarchical recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 5720–5724. IEEE, 2017.

Jean, Sébastien, Cho, Kyunghyun, Memisevic, Roland, and Bengio, Yoshua. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, and Mikolov, Tomas. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Kim, Soo-Min and Hovy, Eduard. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, pp. 1367. Association for Computational Linguistics, 2004.

Le, Quoc and Mikolov, Tomas. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pp. 1188–1196, 2014.

Liang, Dongyun, Xu, Weiran, and Zhao, Yinge. Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 43–47, 2017.

Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Nakov, Preslav and Tiedemann, Jörg. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 301–305. Association for Computational Linguistics, 2012.

Olah, Christopher. Understanding lstm networks. *GITHUB blog, posted on August*, 27:2015, 2015.

Pang, Bo and Lee, Lillian. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, pp. 271. Association for Computational Linguistics, 2004.

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

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In

Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543, 2014.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Santos, Cicero D and Zadrozny, Bianca. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1818–1826, 2014.

Severyn, Aliaksei and Moschitti, Alessandro. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 959–962. ACM, 2015.

Turney, Peter D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 417–424. Association for Computational Linguistics, 2002.

Wilson, Theresa, Wiebe, Janyce, and Hoffmann, Paul. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pp. 347–354. Association for Computational Linguistics, 2005.

Yin, Wenpeng, Kann, Katharina, Yu, Mo, and Schütze, Hinrich. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

Zhang, Xiang, Zhao, Junbo, and LeCun, Yann. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.