Hierarchical Neural Model for Recommending Articles

CS420 Coursework: Text Classification

Runzhe Yang | Yang_Runzhe@sjtu.edu.cn

ACM Honored Class, Shanghai Jiao Tong University

Shanghai Jiao Tong University

Challenge: how to build a Machine Learning Model assisting the human editor in selecting proper articles from a large amount of financial news?

- A Binary Text Classification Task!

Traditional models

bag of words, n-grams and some TFIDF variants

- acceptable performance. ✔
- *hand-crafted* or *rule-generated features* of text. ✘
- restricted expression power and learnability. ✘

Deep learning models

word-based CNN [1, 3] , word-based RNN and character-based CNN [2]

- learn *more flexible features* consistent to the task *automatically*. ✔
- do not take the **hierarchy of document** into account. ✘

Only learns the features and the classification in from the "flat" document representation, which does not conform to to human reading behaviors.

Problem Formulation

Notation	Description
$\mathcal{C}$	The dictionary of Chinese characters, including all special symbols.
$c_1, c_2, \ldots, c_N$	The stream of characters of documents as raw inputs.
$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N$	The corresponding embedded vectors of characters.
$\boldsymbol{f}_\delta^{(k)}$	The feature in k^{th} channel gained by CNN kernel of width δ.
$\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_T$	The compact representation of the document of total length T.
p	The predicted probability $\mathbb{P}(\hat{y} = 1 \vert c_1, c_2, \ldots, c_N)$.

Input: the stream of Chinese characters, $\{c_i\}_N$, with fixed length N=1500.

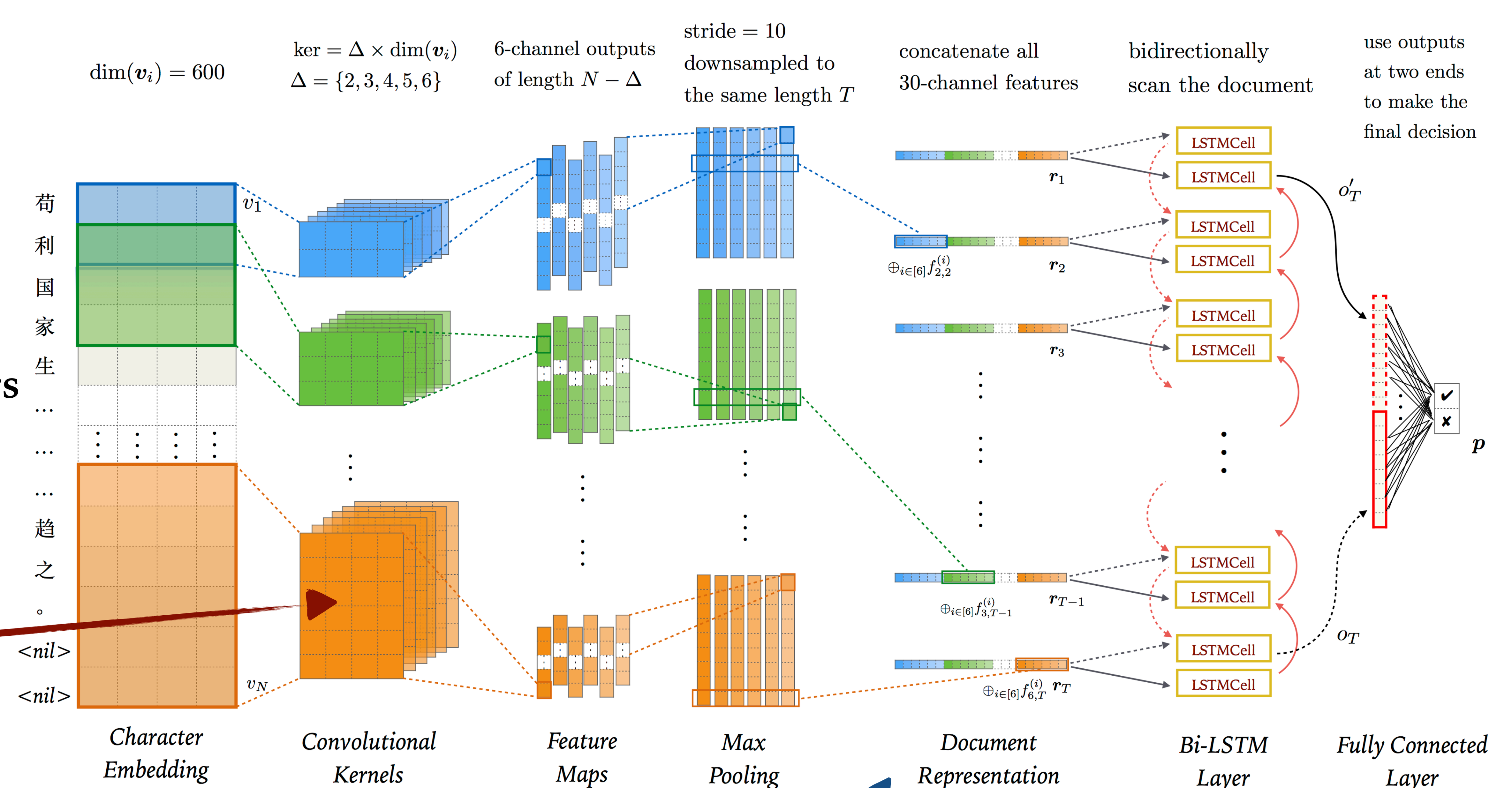
Output: the probability p of acceptance of each document, i.e., $P_\theta(\hat{y} = 1|c_1, c_2, \ldots, c_N)$ where θ are parameters of model.

Cross Entropy Criterion:

$$\mathcal{L}(\theta) = -y \log P_\theta(\hat{y} = 1|c_1, c_2, \ldots, c_N) - \alpha \cdot (1 - y) \log P_\theta(\hat{y} = 0|c_1, c_2, \ldots, c_N)$$

where α ≈ 0.1 is for treating the imbalance issue.

My Solution:
Hierarchical Neural Model



Level 1: Chinese Character Embedding

$$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N = \texttt{embedding}(c_1, c_2, \ldots, c_N)$$

Level 2: CNNs as Word Signal Extractors

- use CNNs with various kernel size to extract the *representation of words and phrases.*

$$\boldsymbol{f}_\delta^{(k)} = \texttt{CNN}_\delta^{(k)}(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T)$$

$$\delta \in \Delta = \{2, 3, 4, 5, 6\}$$

- get *compact representation of document* by concatenating all all 30 channels

$$\boldsymbol{r}_i = \boldsymbol{f}_{2,i}^{(1)} \oplus \cdots \oplus \boldsymbol{f}_{2,i}^{(6)} \oplus \boldsymbol{f}_{3,i}^{(1)} \oplus \cdots \oplus \boldsymbol{f}_{3,i}^{(6)} \oplus \cdots \oplus \boldsymbol{f}_{6,i}^{(1)} \oplus \cdots \oplus \boldsymbol{f}_{6,i}^{(6)}.$$

Level 3: BiLSTM as Document Reader

- use *Bi-directional Long Short Term Memory Networks* since human do not simply read an article from beginning to the end, but *bi-directionally search* for desired information.

$$\begin{cases} \boldsymbol{o}_T = \texttt{LSTM}(\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_T)_T \\ \boldsymbol{o}'_T = \texttt{LSTM}'(\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_T)_T \end{cases} \quad \boldsymbol{p} = \boldsymbol{W}_f(\boldsymbol{o}_T \oplus \boldsymbol{o}'_T) + \boldsymbol{b}_f \quad p = \frac{exp\{\boldsymbol{p}_1\}}{exp\{\boldsymbol{p}_1\} + exp\{\boldsymbol{p}_2\}}$$

Experiments & Results

FastCNN	MultiCNN	CNN+LSTM	Large CNN+LSTM
0.868829	0.869773	0.853858	0.881556
Huge CNN+LSTM	**Huge CNN+LSTM 2**	**Ex CNN+LSTM**	**Proposed HNM**
0.893592	0.894015	0.898972	**0.900659**

Using a two layer neural network of hidden size 256 to stack all the 8 models in this table, I've achieved AUC **0.901014** on the validation set.

Please find the model specification & training settings on https://github.com/RunzheYang/TextClassification

Conclusion:

My proposed Hierarchical Neural Model (HNM) outperforms all the other CNN+LSTM variants. More channels, higher embedding dimensions, and larger LSTM hidden size always benefits the results as our comparison shows. Stacking is an effective way to improve the final performance.