A survey and practice of Neural-network-based Textual representation

WabyWang, LilianWang, JaredWei, LoringLiu
Department of Social Network Operation,
Social Network Group,
Tencent

welcome for any issues and contributions !!!
find . -name "*.py" -print | xargs wc -l

3256 lines
TextZOO
A new Benchmark to Reconsidering Text Classification

Can not do

• Can not directly deploy online
  • Implementing is easy, while design is what really challenging

• Can not tell you the precise hyper-parameter of your task
  • A fish or a fishing skill?

• Can not ensure to improve your performance
  • It depends on the scale of your supervised data
Highly depends on your data and task

- NLP features extraction Model
  - TFIDF is enough strong, e.s. long text
  - A Few pretrained Model
    - Glove/Word2vec only for initialization
    - No common-known CN embedding
    - No pretrained Model

- CV features extraction
  - SIFT or SIFT-like is not very strong.
  - pretrained ResNet from ImageNet

Zero-shot learning can hardly works in NLP, currently
Can do

• Easy to implement a model after talking
  • Talking is cheap, 10 lines a model.

• Directly support all the public dataset
  • Testing model

• Know how to design a DL model for NLP, not only text classification
  • A fishing skill
Contents

• Brief Introduction of TextZoo
• Why text classification?
• General Overview of Text Classification
• Overview of Text Classification in Neural Network approach.
• Architecture of TextZoo
• Conclusions
Contents

• Brief Introduction of TextZoo
• Why text classification?
• General Overview of Text Classification
• Overview of Text Classification in Neural Network approach.
• Architecture of TextZoo
• Conclusions
TextZoo

- Text Classification
  - Sentimental
  - Topic
  - Spam filter
  - ...

- A benchmark
  - 20 Dataset
  - 20 Models

- PyTorch
  - Life is short, I use PyTorch(Python)
Models

✓ FasText
✓ CNN (Kim CNN, Multi-Layer CNN, Multi-perspective CNN, Inception CNN)
✓ LSTM (BILSTM, StackLSTM, LSTM with Attention)
✓ Hybrids between CNN and RNN (RCNN, C-LSTM)
✓ Attention (Self Attention / Quantum Attention)
✓ Transformer - Attention is all you need
✓ Capsule
✓ Quantum-inspired NN
➢ ConS2S
➢ Memory Network
Datasets

- IMDB
- MR
- CR
- MPQA
- SST1
- SST2
- Subj
- TREC
Contents

- Brief Introduction of TextZoo
- Why text classification?
- General Overview of Text Classification
- Overview of Text Classification in Neural Network approach.
- Architecture of TextZoo
- Conclusions
Supervised tasks in NLP

- Classification: assigning a label to a string
  \[ S \rightarrow C \]

- Matching: matching two strings
  \[ S, t \rightarrow \mathbb{R}^+ \]

- Translation: transforming one string to another
  \[ S \rightarrow t \]

- Structured prediction: mapping string to structure
  \[ S \rightarrow s' \]
Why text classification?

Text Representation

Text \rightarrow MLP/CNN/RNN \rightarrow representation \rightarrow classification
Why text classification?

Text Representation

- Text
- MLP/CNN/RNN
- representation
- classification

Text Representation

- Text
- MLP/CNN/RNN
- representation

Matching
Why text classification?

Text Representation

Text → MLP/CNN/RNN → representation → classification

MLP/CNN/RNN → Text
Why text classification?

Text Representation

- Token_1
  - RNN cell
  - representation
  - classification

- Token_2
  - RNN cell
  - representation
  - classification

- Token_3
  - RNN cell
  - representation
  - classification
Examples for LSTM

https://mp.weixin.qq.com/s/MhRrVW44dDX-PpWNqCWCOw
Fundamental Demo In Code with PyTorch pseudo code

- Model = LSTM/CNN/Capsule/...
- text,label = Dataset.nextBatch()
- representation = Model(text)

- Classification = FC(representation)  \( FC : \) Mapping to label size
- Translation = Decode(representation)
- Matching = Cosine(representation1, representation2)
- Sequential Labelling = FCs(representations)
Contents

• Brief Introduction of TextZoo
• Why text classification?
• General Overview of Text Classification
• Overview of Text Classification in Neural Network approach.
• Architecture of TextZoo
• Conclusions
Overview

• Traditional Models
  • Naïve Bayes
  • SVM

• DL Models
  • ???CNN
  • ???RNN
  • ???NN
Traditional Classification

• SVM/Naïve Bayes
  • Bag-of-words(N-gram) hypothesis
  • Features :
    • TFIDF (unigram, N-gram)
    • POS, parser
    • hypernyms, WordNet
    • hand-coded rules
  • May needs “feature selection”
  • Good performance in long text

*It performs better than you expected!!*
Contents

• Brief Introduction of TextZoo
• Why text classification?
• General Overview of Text Classification
• Overview of Text Classification in Neural Network approach.
• Architecture of TextZoo
• Conclusions
Embedding and further DL models

Distributional hypothesis

linguistic items with similar distributions have similar meanings

Localist representation

- **BMW** \([1, 0, 0, 0, 0]\) \([.3, .7, .2, .1, .5]\)
- **Audi** \([0, 0, 0, 1, 0]\) \([.5, .3, .2, .1, .0]\)
- **Benz** \([0, 0, 1, 0, 0]\) \([.2, .0, .31, .03, .01]\)
- **Polo** \([0, 0, 0, 1, 0]\) \([.1, .1, .5, .5, 0.2]\)

## Distributed representation

<table>
<thead>
<tr>
<th>Car</th>
<th>Representation</th>
<th>Size</th>
<th>Color</th>
<th>...</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW</td>
<td>[1, 0, 0, 0, 0, 0]</td>
<td>[.3, .7, .2, .1, .5]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audi</td>
<td>[0, 0, 0, 1, 0]</td>
<td>[.5, .3, .2, .1, .0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benz</td>
<td>[0, 0, 1, 0, 0]</td>
<td>[.2, .0, .31, .03, .01]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polo</td>
<td>[0, 0, 0, 1, 0]</td>
<td>[.1, .1, .5, .5, 0.2]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to get Distributed representation

- **Matrix Factorization**
  - Word-word Matrix
  - Document-word Matrix
    - PLSA
    - LDA
- **Sample-based Prediction**
  - NNLM
  - C & W
  - Word2vec

Glove is a combination between these two schools of approaches
Why embedding is so hot?

• Only automatically build supervised pairs in unsupervised corpus

• *Life is complex. It has both real and imaginary parts*
Word2Vec

The diagram illustrates the architecture of a Word2Vec model. It includes:

- **Softmax classifier**
- **Hidden layer**
- **Projection layer**

The model takes a sentence as input, processes it through layers, and outputs a vector representation for each word. The goal is to learn vector representations that capture the semantic meaning of words.
State-of-art Embedding

• Word2Vec
• Glove
• Many and many improved version of word embedding
  • Improved Word Representation Learning with Sememes
  • “Polysemy problem”
  • “Antonym problem”
  • Complex embedding [We are interested, now]
    • life is complex, it has both real and imaginary parts
Which is the most similar word of “Tencent”? 

May be “Baidu” or “pony”? 

Nie Janyun said in SIGIR 2016 Chinese-Author Workshop, Tsinghua University, Beijing
Attention!!!

*Average Embedding may be a problematic practice for textual representation, especially in long text.*

Should add some *supervised signals* after embedding to reduce the noise !, like *Fastext*

Embedding is everywhere!!!

- Word2vec
- Doc2vec
- Item2vec
- *Everything can be embed!!*

**Embedding** is a kind of approach, while **word vector** is a typical application of embedding.

How to choose Word Vector

• Word2vec or Glove
  • Depends on you final performance, not a prior test in linguistic/syntax regulation

• Embedding dim, depends on scale of training dataset.
  • Larger dataset, bigger dimension, but overfitting.

• If possible, train the embedding on own your data.
  Topic-relevant is somehow more important than the data size
More features in DL

• POS Embedding
• CCG Embedding
• Extract matching Embedding
• Position Embedding

• Embed Every discrete features in Neural Network
  • If it is continuous, bucket it and make it discrete.
MLP
UAT in MLP

Multi-layer Non-linear Mapping $\to$ **Universal Approximation Theorem**
A sample of $\theta(wx+b)$

\[ w = 100 \quad b = -40 \]

\[ \sigma(wx + b), \text{ where } \sigma(z) \equiv 1/(1 + e^{-z}) \]

\[ s = -b/w. \]

An another sample

\[ \sigma(wx + b), \text{ where } \sigma(z) \equiv \frac{1}{1 + e^{-z}} \]
CNN

• Basic CNN
• Kim CNN
• VDCNN
CNN [Kalchbrenner et al. ACL 2014]
CNN [kim EMNLP 2014]

Figure 1: Model architecture with two channels for an example sentence.
Figure 1: Model architecture of fastText for a sentence with $N$ ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.
Why Mr. Lace chooses FasText

• Fast
• Input may a set of keywords instead of a sequential of words
  • (Group name)
• Label may be inaccurate

• Build more hand-code features would get comparable results
Very Large CNN [Conneau EACL ]

Table 4: Best published results from previous work. Zhang et al. (2015) best results use a Thesaurus data augmentation technique (marked with an *). Yang et al. (2016)’s hierarchical methods is particularly adapted to datasets whose samples contain multiple sentences.

<table>
<thead>
<tr>
<th>Depth</th>
<th>Pooling</th>
<th>AG</th>
<th>Sogou</th>
<th>DBP</th>
<th>Yelp P</th>
<th>Yelp F</th>
<th>Yah. A</th>
<th>Amz. F</th>
<th>Amz. P</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Convolution</td>
<td>10.17</td>
<td>4.22</td>
<td>1.64</td>
<td>5.01</td>
<td>37.63</td>
<td>38.10</td>
<td>38.52</td>
<td>4.94</td>
</tr>
<tr>
<td>9</td>
<td>KMaxPooling</td>
<td>9.83</td>
<td>3.58</td>
<td>1.56</td>
<td>5.27</td>
<td>38.04</td>
<td>38.24</td>
<td>39.19</td>
<td>5.69</td>
</tr>
<tr>
<td>9</td>
<td>MaxPooling</td>
<td>9.17</td>
<td>3.70</td>
<td>1.35</td>
<td>4.88</td>
<td>36.73</td>
<td>37.95</td>
<td>37.95</td>
<td>4.70</td>
</tr>
<tr>
<td>17</td>
<td>Convolution</td>
<td>9.29</td>
<td>3.94</td>
<td>1.42</td>
<td>4.96</td>
<td>36.10</td>
<td>37.35</td>
<td>37.50</td>
<td>4.53</td>
</tr>
<tr>
<td>17</td>
<td>KMaxPooling</td>
<td>9.39</td>
<td>3.51</td>
<td>1.61</td>
<td>5.05</td>
<td>37.41</td>
<td>38.25</td>
<td>38.81</td>
<td>5.43</td>
</tr>
<tr>
<td>17</td>
<td>MaxPooling</td>
<td>8.88</td>
<td>3.54</td>
<td>1.40</td>
<td>4.50</td>
<td>36.07</td>
<td>37.51</td>
<td>37.39</td>
<td>4.41</td>
</tr>
<tr>
<td>29</td>
<td>Convolution</td>
<td>9.36</td>
<td>3.61</td>
<td>1.36</td>
<td>4.35</td>
<td>35.28</td>
<td>37.17</td>
<td>37.58</td>
<td>4.28</td>
</tr>
<tr>
<td>29</td>
<td>KMaxPooling</td>
<td>8.67</td>
<td>3.18</td>
<td>1.41</td>
<td>4.63</td>
<td>37.00</td>
<td>37.16</td>
<td>38.39</td>
<td>4.94</td>
</tr>
<tr>
<td>29</td>
<td>MaxPooling</td>
<td>8.73</td>
<td>3.36</td>
<td>1.29</td>
<td>4.28</td>
<td>35.74</td>
<td>37.57</td>
<td>37.00</td>
<td>4.31</td>
</tr>
</tbody>
</table>

Table 5: Testing error of our models on the 8 data sets. No data preprocessing or augmentation is used.

Figure 1: VDCNN architecture.
Go deeper or not?

**DEEP**
- Slower
- Overfitting
  - More Parameters, more data need to feed
- Hard for convergence
  - Highway network
  - Residual Block
  - Inception

**Shallow: one-layer**
- Fast
- Less data, es. Fastext
Go deeper or not?

Image recognition: Pixel $\rightarrow$ edge $\rightarrow$ texton $\rightarrow$ motif $\rightarrow$ part $\rightarrow$ object

Text: Character $\rightarrow$ word $\rightarrow$ word group $\rightarrow$ clause $\rightarrow$ sentence $\rightarrow$ story

Speech: Sample $\rightarrow$ spectral band $\rightarrow$ sound $\rightarrow$ ... $\rightarrow$ phone $\rightarrow$ phoneme $\rightarrow$ word

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Modified from Prof. LeCun and Prof. Bengio
RNN and its Variant

- RNN
- LSTM
- LSTM + mean
- LSTM + bidirectional
- LSTM + Attention
- LSTM + Stack
- LSTM + Self-Attention
- TreeLSTM
Bias in RNN
Bias in RNN
From RNN to LSTM

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
• How many gates?
• Difference between cell and the hidden state?
• How many parameters in a LSTM?
Forget gate

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]
Input gate

\[ i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \]

\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]

replace tanh with softsign (not softmax) activation for prevent overfitting

https://zhuanlan.zhihu.com/p/21952042
Forgotten + input

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]
Output Gate

\[ o_t = \sigma (W_o [h_{t-1}, x_t] + b_o) \]
\[ h_t = o_t \ast \tanh (C_t) \]
LSTM Variants: Peephole connections

\[
f_t = \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)
\]
\[
i_t = \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)
\]
\[
o_t = \sigma (W_o \cdot [C_t, h_{t-1}, x_t] + b_o)
\]
**LSTM Variants:** coupled forget and input gates

\[ C_t = f_t \cdot C_{t-1} + (1 - f_t) \cdot \tilde{C}_t \]
LSTM Variants: GRU

$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$

$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$

$\tilde{h}_t = \tanh (W \cdot [r_t \cdot h_{t-1}, x_t])$

$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$

✓ Hidden = Cell
✓ Forget gate + input gate =1
BiLSTM

Output Layer

Backward Layer

Forward Layer

Input Layer
Last or Mean?
RNN/LSTM with Attention

https://www.jianshu.com/p/4fbc4939509f
Visualization of Attention in RNN/LSTM

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

A large white bird standing in a forest.
A woman holding a clock in her hand.
A man wearing a hat and a hat on a skateboard.
A person is standing on a beach with a surfboard.
A woman is sitting at a table with a large pizza.
A man is talking on his cell phone while another man watches.
Visualization of Attention in RNN/LSTM

(a) Hypothesis: A boy is riding an animal.
(b) Hypothesis: A girl is wearing a blue jacket.
(c) Hypothesis: Two dogs swim in the lake.
(d) Hypothesis: Two mimes sit in complete silence.

Sematic Entailment

Speech Recognition
Deeper LSTM
Deeper LSTM

Deep is not necessary, but more data!!!
### Comparative Study of CNN and RNN for Natural Language Processing

<table>
<thead>
<tr>
<th>TextC</th>
<th>performance</th>
<th>lr</th>
<th>hidden</th>
<th>batch</th>
<th>semLen</th>
<th>filter</th>
<th>size</th>
<th>margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiC (acc)</td>
<td>CNN 82.38</td>
<td>0.2</td>
<td>20</td>
<td>5</td>
<td>60</td>
<td>3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRU 86.32</td>
<td>0.1</td>
<td>30</td>
<td>50</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 84.51</td>
<td>0.2</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RC (F1)</td>
<td>CNN 68.02</td>
<td>0.12</td>
<td>70</td>
<td>10</td>
<td>20</td>
<td>3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRU 68.56</td>
<td>0.12</td>
<td>80</td>
<td>100</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 66.45</td>
<td>0.1</td>
<td>80</td>
<td>20</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TE (acc)</td>
<td>CNN 77.13</td>
<td>0.1</td>
<td>70</td>
<td>50</td>
<td>50</td>
<td>3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRU 78.78</td>
<td>0.1</td>
<td>50</td>
<td>80</td>
<td>65</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 77.85</td>
<td>0.1</td>
<td>80</td>
<td>50</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SemMatch</td>
<td>CNN (63.69, 65.01)</td>
<td>0.01</td>
<td>30</td>
<td>60</td>
<td>40</td>
<td>3</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AS (MAP &amp; MRR)</td>
<td>GRU (62.58, 63.59)</td>
<td>0.1</td>
<td>80</td>
<td>150</td>
<td>40</td>
<td>-</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>LSTM (62.00, 63.26)</td>
<td>0.1</td>
<td>60</td>
<td>150</td>
<td>45</td>
<td>-</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>QRM (acc)</td>
<td>CNN 71.50</td>
<td>0.125</td>
<td>400</td>
<td>50</td>
<td>17</td>
<td>5</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRU 69.80</td>
<td>1.0</td>
<td>400</td>
<td>50</td>
<td>17</td>
<td>-</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 71.44</td>
<td>1.0</td>
<td>200</td>
<td>50</td>
<td>17</td>
<td>-</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>SeqOrder</td>
<td>CNN 54.42</td>
<td>0.01</td>
<td>250</td>
<td>50</td>
<td>5</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRU 55.67</td>
<td>0.1</td>
<td>250</td>
<td>50</td>
<td>5</td>
<td>-</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 55.39</td>
<td>0.1</td>
<td>300</td>
<td>50</td>
<td>5</td>
<td>-</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>ContextDep</td>
<td>CNN 94.18</td>
<td>0.1</td>
<td>100</td>
<td>10</td>
<td>60</td>
<td>5</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>POS tagging (acc)</td>
<td>GRU 93.15</td>
<td>0.1</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSTM 93.18</td>
<td>0.1</td>
<td>200</td>
<td>70</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bi-GRU 94.26</td>
<td>0.1</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bi-LSTM 94.35</td>
<td>0.1</td>
<td>150</td>
<td>5</td>
<td>60</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
RNN vs CNN

- RNN
  - 序列结构
  - 强调高阶关系
  - 位置跳跃的依赖

- 速度
  - 更慢，串行
  - 方便定长，通过attention

- CNN
  - 两个句子关系
  - N-gram匹配更重要的match场景
  - 局部依赖关系

- 速度
  - 可以并行，更灵活
  - 输出不定长，跟文本长度有关
CNN vs RNN vs their Hybrids

<table>
<thead>
<tr>
<th>Neural Network Model</th>
<th>Avg. Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed-Forward (Word Embeddings) [1]</td>
<td>58.4%</td>
</tr>
<tr>
<td>Feed-Forward (Feature Vectors) [1]</td>
<td>66.8%</td>
</tr>
<tr>
<td>CNN</td>
<td>66.7%</td>
</tr>
<tr>
<td>LSTM</td>
<td>72.5%</td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>69.7%</td>
</tr>
<tr>
<td>LSTM-CNN</td>
<td>75.2%</td>
</tr>
</tbody>
</table>

http://blog.csdn.net/youngair/article/details/78013352

Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model
From a Industrial perspective

• Add **features**.
• **Understanding** your data: pay more attention on data **preparation**.
• **Parameter** adjusting with a robust setting
  • *Oh, overfit*
• **Model** is not very important, especially data is not low-quality.
  • *Models differs slightly in low-quality data.*
• Trade-off between performance and **efficiency**
  • *For example, multi-size kennels is better but slower!*
Related Models

• Do not directly aims at this task, but also aims to build a text representation.
  • ConvS2S
  • Attention is all you need
  • Dynamic Memory Network
Conv S2S
Attention is all you need

Figure 1: The Transformer - model architecture.
Self-Attention

Figure 1: A sample model structure showing the sentence embedding model combined with a fully connected and softmax layer for sentiment analysis (a). The sentence embedding $M$ is computed as multiple weighted sums of hidden states from a bidirectional LSTM $(h_1, \ldots, h_n)$, where the summation weights $(A_{11}, \ldots, A_{1n})$ are computed in a way illustrated in (b). Blue colored shapes stand for hidden representations, and red colored shapes stand for weights, annotations, or input/output.
Dynamic Memory Network
Other models

• Tree-LSTM
• Pointer networks
• Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling (T. Shen et al., ICLR 2018)
• Directional Self-Attention Network
• Recurrent Entity Network
Char-CNN

Component-Enhanced Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. Component-Enhanced Chinese Character Embeddings. Proceedings of EMNLP, 201
Char-word Hybrids

Combining Word-Level and Character-Level Representations for Relation Classification of Informal Text
Long text/document classification

- Hierarchical Attention Networks (HAN)
Multi-task Learning

Adversarial Multi-task Learning

RL for text classification

• Learning Structured Representation for Text Classification via Reinforcement Learning  AAAI 2018 minlieHuang

<table>
<thead>
<tr>
<th>Models</th>
<th>MR</th>
<th>SST</th>
<th>Subj</th>
<th>AG</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>77.4*</td>
<td>46.4*</td>
<td>92.2</td>
<td>90.9</td>
</tr>
<tr>
<td>biLSTM</td>
<td>79.7*</td>
<td>49.1*</td>
<td>92.8</td>
<td>91.6</td>
</tr>
<tr>
<td>CNN</td>
<td>81.5*</td>
<td>48.0*</td>
<td>93.4*</td>
<td>91.6</td>
</tr>
<tr>
<td>RAE</td>
<td>76.2*</td>
<td>47.8</td>
<td>92.8</td>
<td>90.3</td>
</tr>
<tr>
<td>Tree-LSTM</td>
<td>80.7*</td>
<td>50.1</td>
<td>93.2</td>
<td>91.8</td>
</tr>
<tr>
<td>Self-Attentive</td>
<td>80.1</td>
<td>47.2</td>
<td>92.5</td>
<td>91.1</td>
</tr>
<tr>
<td>ID-LSTM</td>
<td>81.6</td>
<td>50.0</td>
<td>93.5</td>
<td>92.2</td>
</tr>
<tr>
<td>HS-LSTM</td>
<td>82.1</td>
<td>49.8</td>
<td>93.7</td>
<td>92.5</td>
</tr>
</tbody>
</table>
Adversarial Training Methods For Semi-supervised Text Classification

Table 2: Test performance on the IMDB sentiment classification task. * indicates using pretrained embeddings of CNN and bidirectional LSTM.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (without embedding normalization)</td>
<td>7.33%</td>
</tr>
<tr>
<td>Baseline</td>
<td>7.39%</td>
</tr>
<tr>
<td>Random perturbation with labeled examples</td>
<td>7.20%</td>
</tr>
<tr>
<td>Random perturbation with labeled and unlabeled examples</td>
<td>6.78%</td>
</tr>
<tr>
<td>Adversarial</td>
<td>6.21%</td>
</tr>
<tr>
<td>Virtual Adversarial</td>
<td>5.91%</td>
</tr>
<tr>
<td>Adversarial + Virtual Adversarial</td>
<td>6.09%</td>
</tr>
<tr>
<td>Virtual Adversarial (on bidirectional LSTM)</td>
<td>5.91%</td>
</tr>
<tr>
<td>Adversarial + Virtual Adversarial (on bidirectional LSTM)</td>
<td>6.02%</td>
</tr>
<tr>
<td>Full+Unlabeled+BoW (Maas et al., 2011)</td>
<td>11.11%</td>
</tr>
<tr>
<td>Transductive SVM (Johnson &amp; Zhang, 2015b)</td>
<td>9.99%</td>
</tr>
<tr>
<td>NBSVM-bigrams (Wang &amp; Manning, 2012)</td>
<td>8.78%</td>
</tr>
<tr>
<td>Paragraph Vectors (Le &amp; Mikolov, 2014)</td>
<td>7.42%</td>
</tr>
<tr>
<td>SA-LSTM (Dai &amp; Le, 2015)</td>
<td>7.24%</td>
</tr>
<tr>
<td>One-hot bi-LSTM* (Johnson &amp; Zhang, 2016b)</td>
<td>5.94%</td>
</tr>
</tbody>
</table>

• ICLR 2017
To-do List

• Support more datasets, especially in Chinese
• Support more models
• Fine-tune the result.
• Installable Library with Python (Pip)