Text Classification Using Deep Neural Networks and Logistic Regression

Haotian Ye Ercong Nie Han-Ching Chen

Center for Information and Language Processing

LMU Munich

Dec 14, 2020

Outline

Text Classification Using with fastText

- **fastText:**[1]
  - free library from Facebook AI Research[2]
  - text classification[3], word vector representation
  - an extension[4] of skipgram word2vec
  - trains models significantly faster[5] then other libraries
  - computes embeddings for character ngrams
  - Models for language identification and various supervised tasks

[1] https://fasttext.cc/
[2] https://research.fb.com/blog/2016/08/fasttext/
[3] An Example: Cooking:https://fasttext.cc/docs/en/supervised-tutorial.html
[4] See slide 5. and 6.
[5] Joulin et al. (2016)

fastText vs. Word2Vec

Limitation of Word2Vec:

- Out of Vocabulary Words:
  - tensor,flow,tensorflow
- Morphology (shared radical):
  - eat,eats,eaten,eater,eating

fastText vs. Word2Vec

Improvement by fastText[6]

- The key insight was to use the internal structure of a word to improve vector representations obtained from the skip-gram method.

[6] Bojanowski et al. (2017)

Neural Network Classification vs. Other Classifier Algorithms

- **Strengths:**
  - high dimensionality problems
  - complex relations between variables
  - classify and label images, audio, and video
  - perform sentiment analysis on text
  - classify security incidents into risk categories
- **Weaknesses:**
  - complex
  - difficult to implement
  - careful fine-tuning

Dataset and Embeddings

- “letters-dataset” for author and language identification
- pre-trained multilingual embeddings[7]

[7] https://github.com/facebookresearch/MUSE

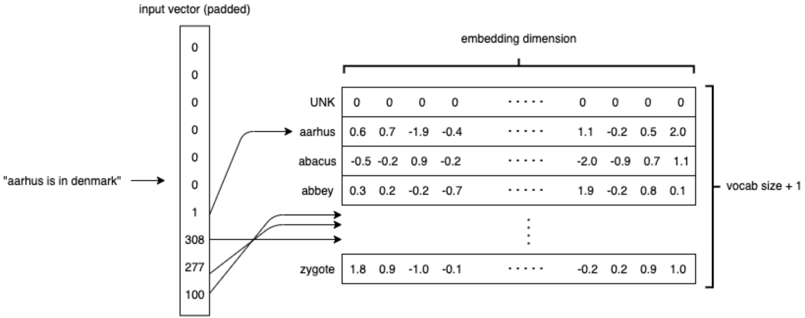
Data Preprocessing

- extraction of useful information (text, author, language)
- removal of noisy data
  - texts in hu and sv
  - lang tag marked as unknown
- encoding of authors/languages as categorical labels
- random sampling to create a dev set (80/10/10 split)

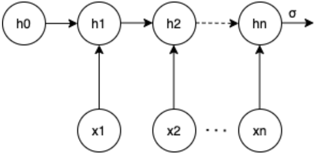
Neural Networks - Key Concepts

- layers
- weight matrices W
- input vector to layer x
- output vector $y = \sigma(Wx + b)$
- activation functions
- backpropagation

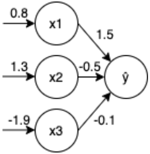
Neural Networks - Lookup Layer



Neural Networks - Dense & Recurrent Layers

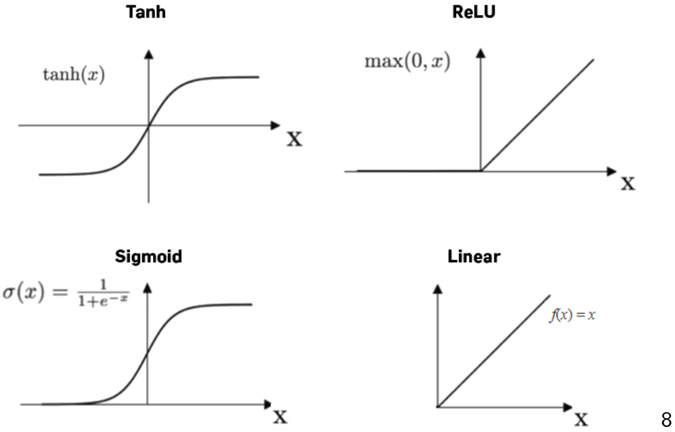


- each input "time step" concatenated with previous hidden state



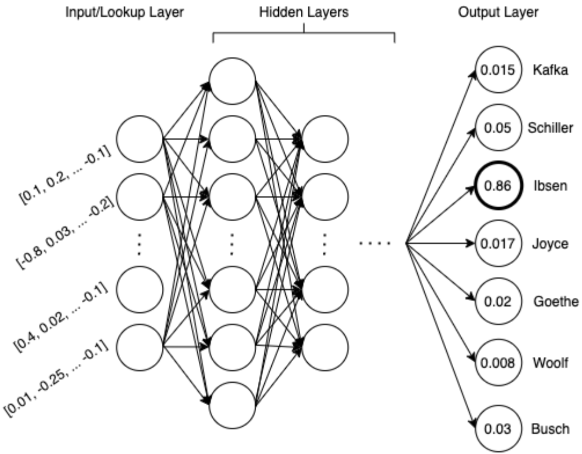
- regular fully connected layer
- layer output fed into the next layer

Activation Functions



[8] https://docs.paperspace.com/machine-learning/wiki/activation-function

Model Illustration



Keras

- Python library for convenient model building (Chollet, 2017)
- `Sequential` vs `Functional`

Keras - Model Construction Steps

- import necessary components from keras
- model initialization

```
model = Sequential()
```

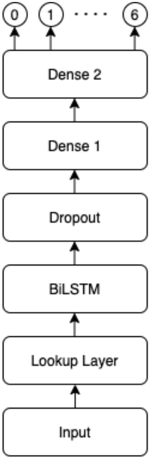
- adding layers

```
model.add(Dense(128, activation='relu'))
```

- model compilation

```
model.compile(optimizer='...', loss='...', metrics=['acc'])
```

Author Classification - Model Architecture

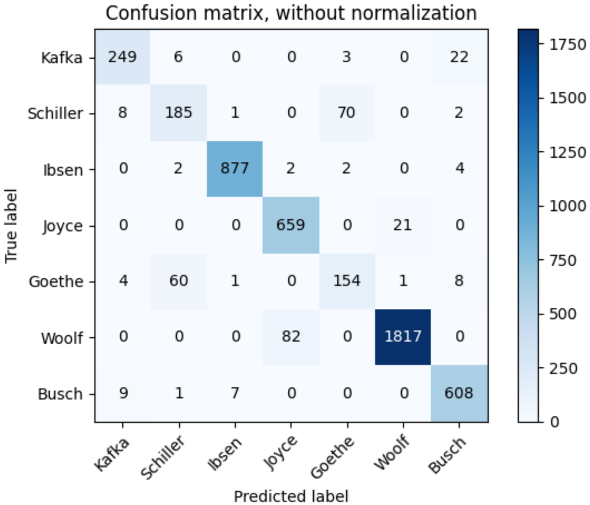


- 300-dim embedding layer
- BiLSTM (**64**/128/156)
- dropout (0.2/**0.4**/0.6)
- dense layer(s) (best w/ **2*128**)

Author Classification - Other Hyperparameters Considered

- `maxlen` (**75** (covers 90%) / 54 (avg))
- EmbLayer initialization
  - random
  - pre-trained (frozen)
  - **pre-trained (updated)**
- Detailed results in the appendix

Author Classification - Results



Author Classification - Results

classification report

	Precision	Recall	F1
Kafka	0.92	0.89	0.91
Schiller	0.73	0.70	0.71
Ibsen	0.99	0.99	0.99
Joyce	0.89	0.97	0.93
Goethe	0.67	0.68	0.67
Woolf	0.99	0.96	0.97
Busch	0.94	0.97	0.96
accuracy			**0.94**
macro avg	0.88	0.88	**0.88**
weighted avg	0.94	0.94	0.94

Language Detection

- classifier - logistic regression
- python package: scikit-learn
- results

Why Logistic Regression?

- Neural Network can explore complex semantic meanings with multiple layers.
- Language Detection is a task to analyze word forms.
- **Overview of Logistic Regression**
  - a discriminative classifier model
  - multiple linear regressions
  - calculate the probability distribution $P(class|features)$
  - other names: Maximum Entropy Classifier, Logit-Model, Log-linear Model

What is Logistic Regression?

- **First step** (Regression): For each instance $\boldsymbol{x}_i$, predict a score z_{ij} for every possible class c_j

$$z_{ij} = \boldsymbol{x}_i \cdot \boldsymbol{w}_j$$

Each feature has a corresponding weight to each class.

- **Second step** (Logistic): Re-scaling and normalization
Turn scores z into the probability distribution

$$P(Y = j|\boldsymbol{x}, \boldsymbol{W}) = \frac{exp(z_j)}{\sum_{j=1}^{k} exp(z_j)}$$

Logistic Regression - Algorithm (1)

- **Input data:**
  Design matrix $\boldsymbol{X}$ of shape *num_instance* $\times$ *num_features*
  Label vector $\boldsymbol{y}$
- **Parameter:** the feature weights matrix $\boldsymbol{W}$ with the size of $k \times m$

$$\boldsymbol{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix}$$

Logistic Regression - Algorithm (2)

- **Parameter learning method:** Maximum Likelihood

$$Likelihood(\boldsymbol{W}) = P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{W}) = \prod_{i=1}^{n} P(Y = y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{W})$$

  $Likelihood(\boldsymbol{W})$ is the product of all probabilities of true labels. The goal is to find the feature weights matrix $\boldsymbol{W}$ that can maximize the likelihood.

- **Optimization method:** gradient descent
- **Hyper parameters:**
  - N-gram (Number of grams) **3**
  - Number of features **all**
  - Regularization: to avoid over-fitting
    - Regularization type ($L1$/**L2**)
    - Regularization intensity C **0.01**

Logistic Regression in Scikit-Learn

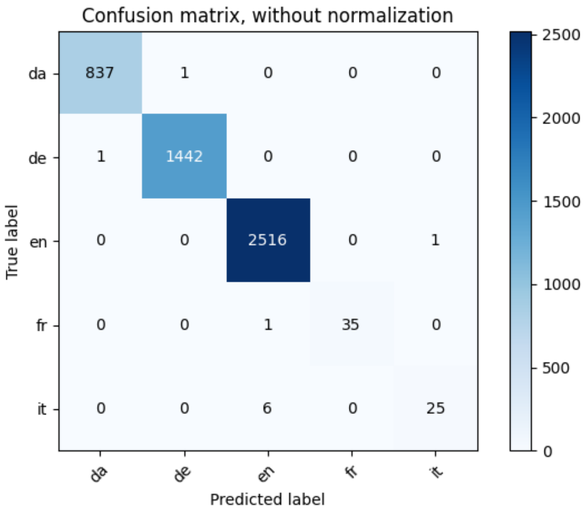
- **Vectorizer**

```
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

- **LogisticRegression Model**

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty = 'l2', C=0.01)
```

Language Detection - Results



Language Detection - Results

classification report

	Precision	Recall	F1	Support
da	1.00	1.00	1.00	838
de	1.00	1.00	1.00	1443
en	1.00	1.00	1.00	2517
fr	1.00	0.97	0.99	36
it	0.96	0.81	0.88	31
accuracy			**1.00**	4865
macro avg	0.99	0.96	**0.97**	4865
weighted avg	1.00	1.00	1.00	

Bibliography

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information.

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification.

Author Classification - Detailed Hyperparameter Search

BiLSTM	Dropout	Dense	MAXLEN	Micro-F1	Macro-F1
Dropout					
64	0.2	1*64	75	0.9366	0.8745
64	0.4	1*64	75	**0.9389**	**0.8782**
64	0.6	1*64	75	0.9308	0.8598
BiLSTM					
128	0.2	1*64	75	0.9303	0.8552
128	0.4	1*64	75	0.9339	0.8671
256	0.4	1*64	75	0.9346	0.8592
256	0.6	1*64	75	0.9279	0.8560
Dense					
64	0.4	2*64	75	0.9317	0.8582
64	0.4	2*128	75	**0.9400**	**0.8855**
64	0.4	3*64	75	0.9206	0.8161
64	0.4	3*128	75	0.9308	0.8561
128	0.4	2*64	75	0.9300	0.8593
128	0.4	2*128	75	0.9349	0.8615

Author Classification - Detailed Hyperparameter Search

Padding Length (MAXLEN)[9]					
BiLSTM	Dropout	Dense	MAXLEN	Micro-F1	Macro-F1
64	0.4	2*128	54	0.9306	0.8644
64	0.4	1*64	54	0.9340	0.8746
Pre-Trained Embeddings (frozen)					
64	0.4	2*128	75	0.8804	0.7656
64	0.4	1*64	75	0.8835	0.7753
Pre-Trained Embeddings (updated)					
BiLSTM	Dropout	Dense	MAXLEN	Micro-F1	Macro-F1
64	0.4	2*128	75	**0.9416**	**0.8864**[10]
64	0.4	1*64	75	0.9351	0.8631

[9] 2 boldfaced settings from previous results used to further evaluate effectiveness of a shorter padding length and pre-trained embeddings

[10] setting used to generate confusion matrix and classification report