Tutorial on Deep Learning for Natural Language Processing
ICON-2017, Jadavpur University, Kolkata, India.

Sentiment Analysis

Md Shad Akhtar
Research Scholar
AI-NLP-ML Group
Department of Computer Science & Engineering
Indian Institute of Technology Patna

shad.pcs15@iitp.ac.in
https://iitp.ac.in/~shad.pcs15/

Sentiment Analysis

- Sentiment analysis aims to identify the orientation of opinion in a piece of text.

Why do we need Sentiment Analysis?

- *What others think* has always been an important piece of information.
- *Overwhelming amounts of information* on one topic: Manually reading or analysing all data is very inefficient.
- *Baised/Fake* reviews
- An example
  - Mr. X needs to buy a phone. He was browsing amazon.in and found 1000 reviews for a particular phone.

Scenario 1:

- Let there are ***850 negative***, ***100 positive*** and ***50 neutral*** reviews
- Sentiment → **Negative**.

What if all the 100 positive reviews are at the top?

Scenario 2:

- Let there are ***420 negative***, ***480 positive*** and ***100 neutral*** reviews.
- Sentiment → **Positive**

What if few of the reviews (e.g. 100) are fake?

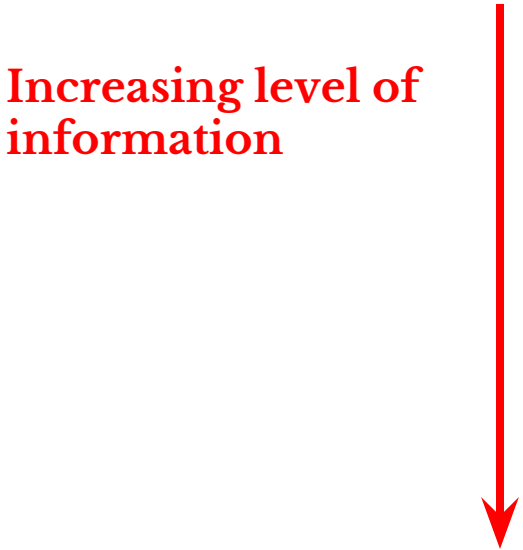
Challenges

- Similar lexical features but different sentiments
  - **यह मूवी अच्छी नहीं है। (This movie is not good.)**
  - **कोई भी मूवी इस से अच्छी नहीं हो सकती। (No movie can be better than this.)**

- Different style of writing but same sentiment
  - **सैमसंग का फ़ोन बहुत ही बेकार है। (Samsung phone is extremely useless.)**
  - **सैमसंग फ़ोन पर मेरे पैसे बरबाद हो गए। (My money was wasted on Samsung phone.)**
  - **सैमसंग से अच्छा मैं आईफ़ोन खरीद लेता। (I could have bought Iphone instead of Samsung.}**

Levels of sentiment analysis

- **Document level**
  - Overall sentiment of a document
  - A document consists of many sentences/paragraphs
- **Sentence level**
  - Sentiment of a stand alone sentence
- **Phrase level**
  - Sentiment towards a given phrase in a sentence
- **Aspect level**
  - Sentiment towards an attribute/feature/aspect of a sentence.
  - Aspect is an attribute or component of the product that has been commented on in a review
  - ***Sentiment targets*** helps us to understand the sentiment analysis problem better.



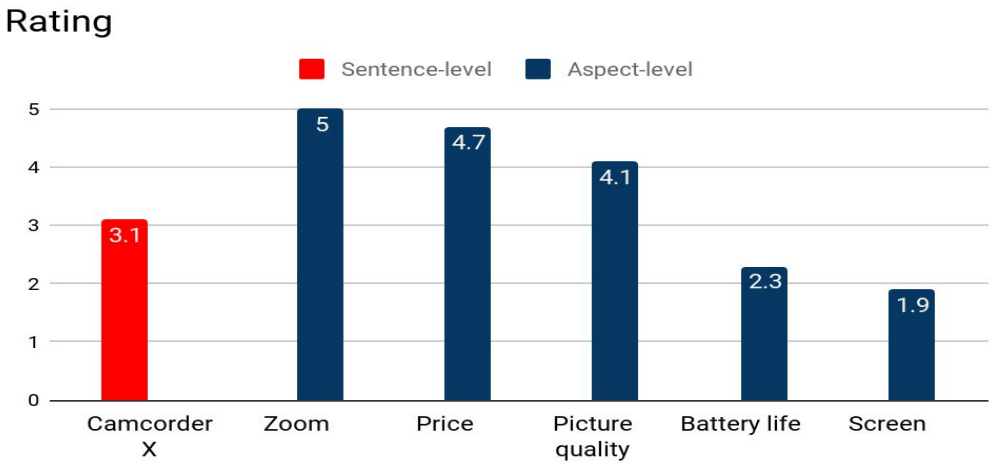
Sentence level v/s Aspect level

- **Sentence level**
  - **इसकी बैटरी शानदार है, लेकिन कैमरा बहुत ही ख़राब है। (Its battery is awesome but camera is very poor.)**
  - **Sentiment: Both *positive* and *negative* → *conflict***

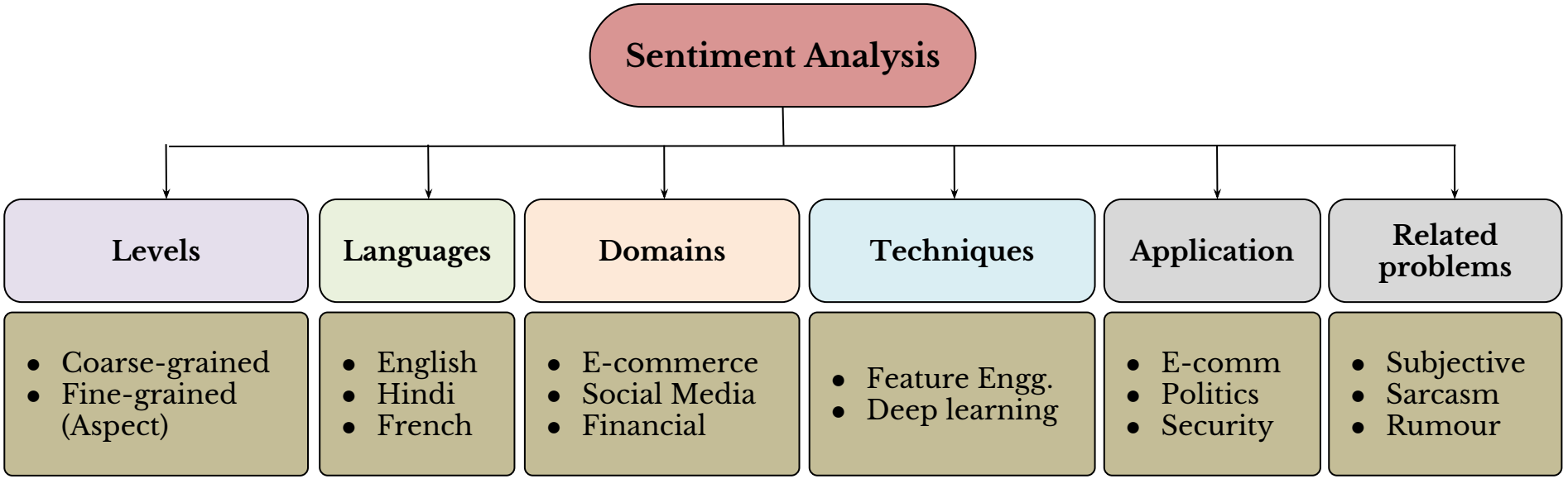
- **Aspect level**
  - **इसकी बैटरी शानदार है, लेकिन कैमरा बहुत ही ख़राब है। (Its battery is awesome but camera is very poor.)**
  - **Aspect terms and their sentiments:**
    - **बैटरी (battery) → positive**
    - **कैमरा (camera) → negative**

Sentence level v/s Aspect level

- **Informed decision**
  - **Camcorder X**
    - The zoom is excellent, but the LCD is blurry.
    - Great value for the price.
    - Although the display is poor the picture quality is amazing.
    - Batteries drain pretty quickly.
    - I love this camera but for short battery life is definitely a pain.

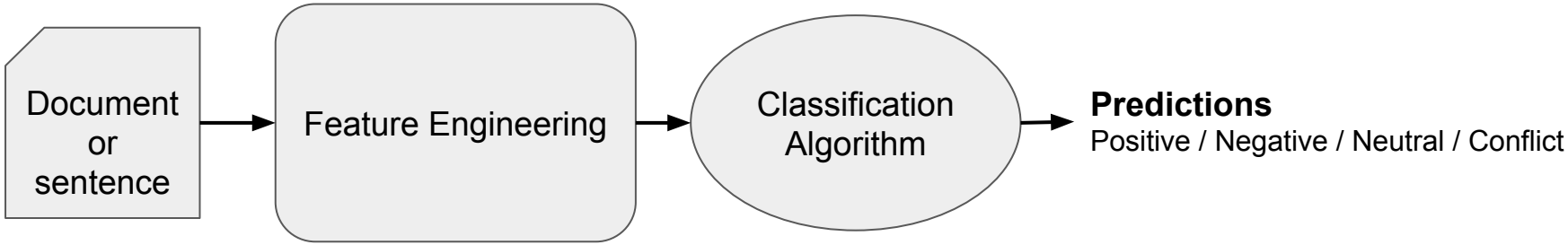


Sentiment Analysis: A boarder view



Pre-deep learning approaches for Sentiment Analysis (Traditional ML approaches)

A traditional classification pipeline

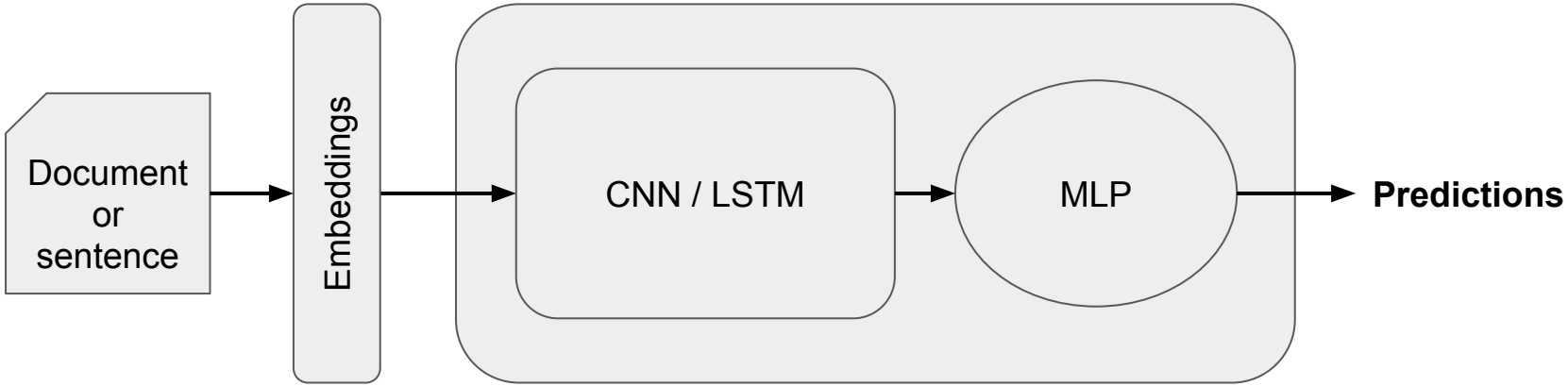


- Ngrams
- Presence or Absence of cue words
- Lexicons

- SVM
- Decision Tree

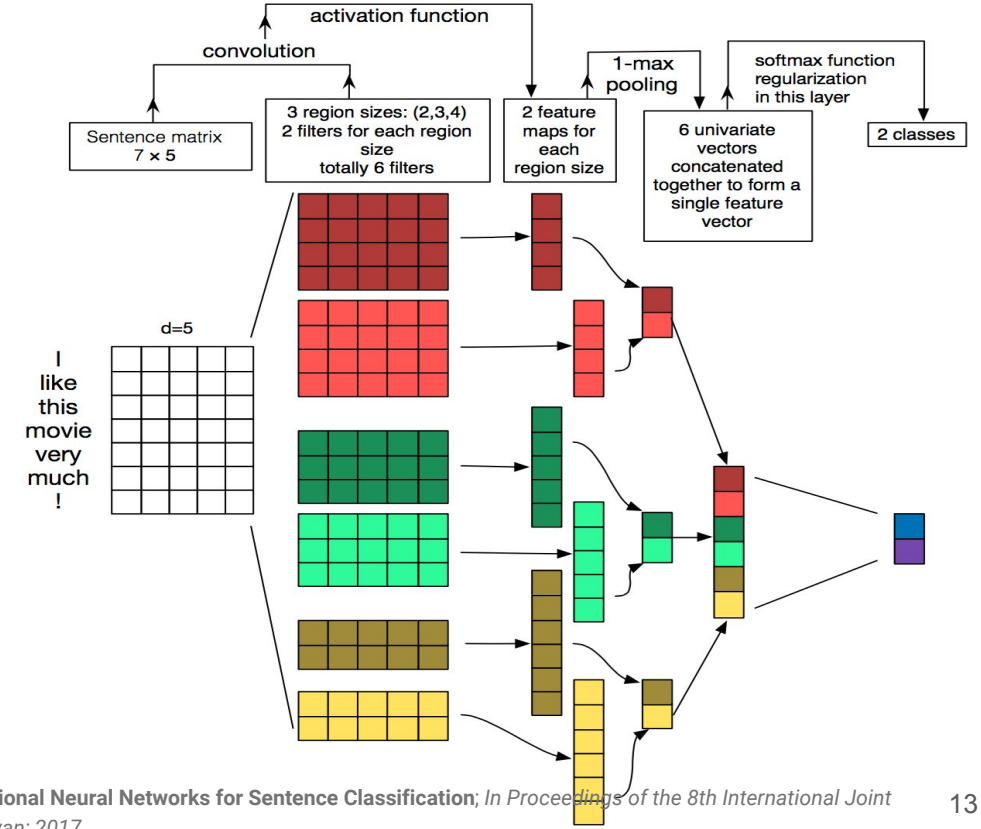
Deep Architectures for Sentiment Analysis

A typical deep learning solution



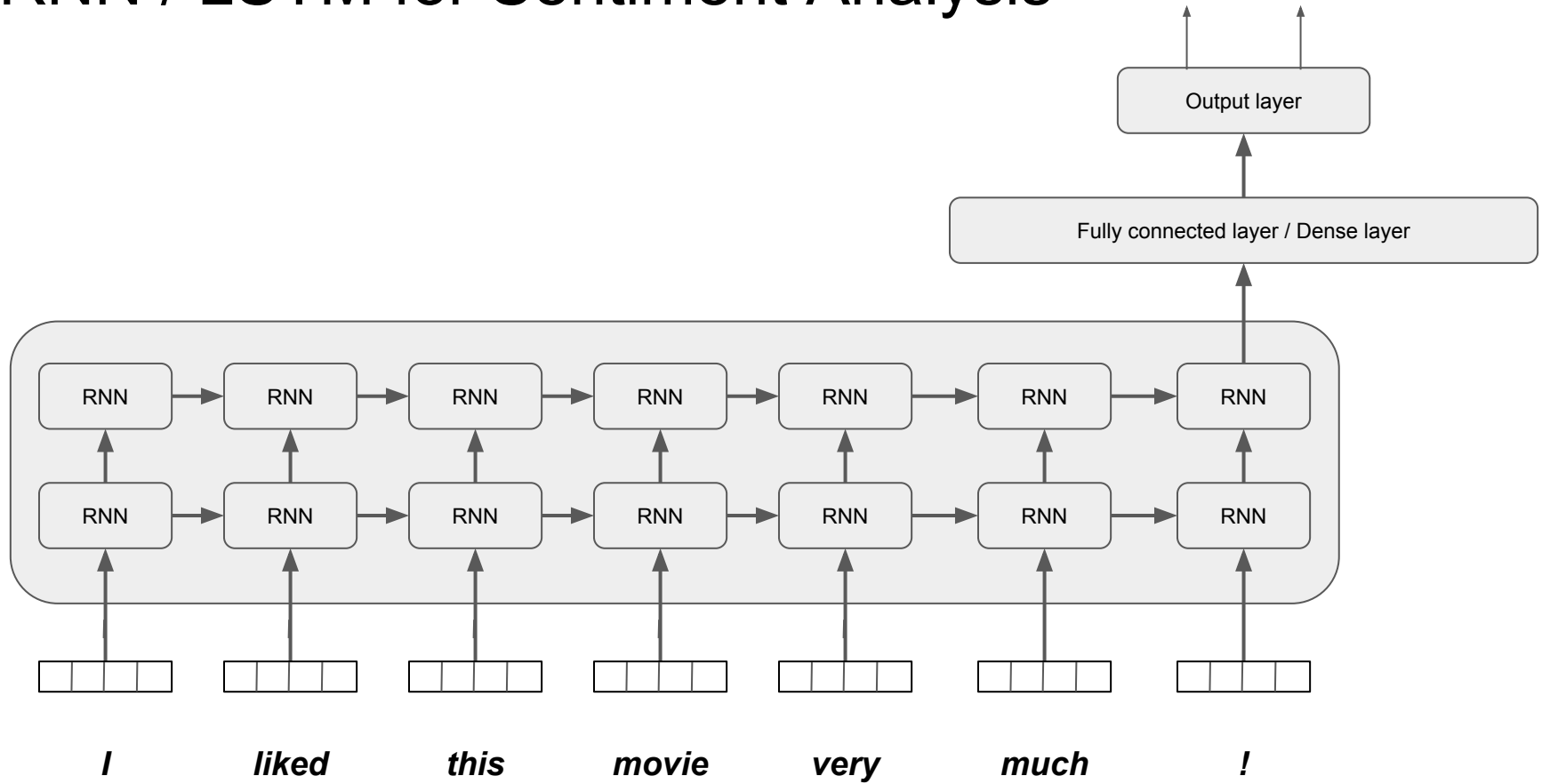
A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

1. Sentence matrix
   a. embeddings of words
2. Convolution filters
   a. Total 6 filters; Two each of size 2, 3 & 4.
   b. 6 feature maps for each filter
3. Pooling
   a. 1-max pooling
4. Concatenate the max-pooled vector
5. Classification
   a. Softmax



Zhang, Y., Wallace, B. ; A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification; *In Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP-2017); pages 253-263; Taipie, Taiwan; 2017.*

RNN / LSTM for Sentiment Analysis



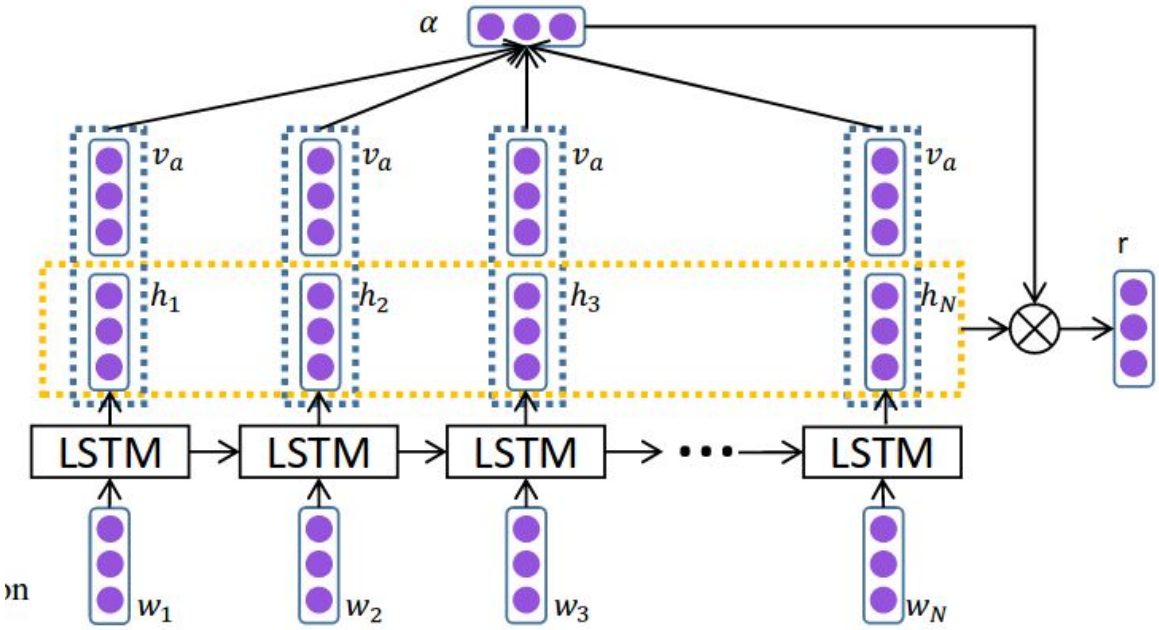
Attention-based LSTM for Aspect-level Sentiment Classification

Staffs are not that friendly, but the taste covers all.

- Aspects / Aspect Category
  - ***Service*** → *negative*
  - ***Food*** → *positive*

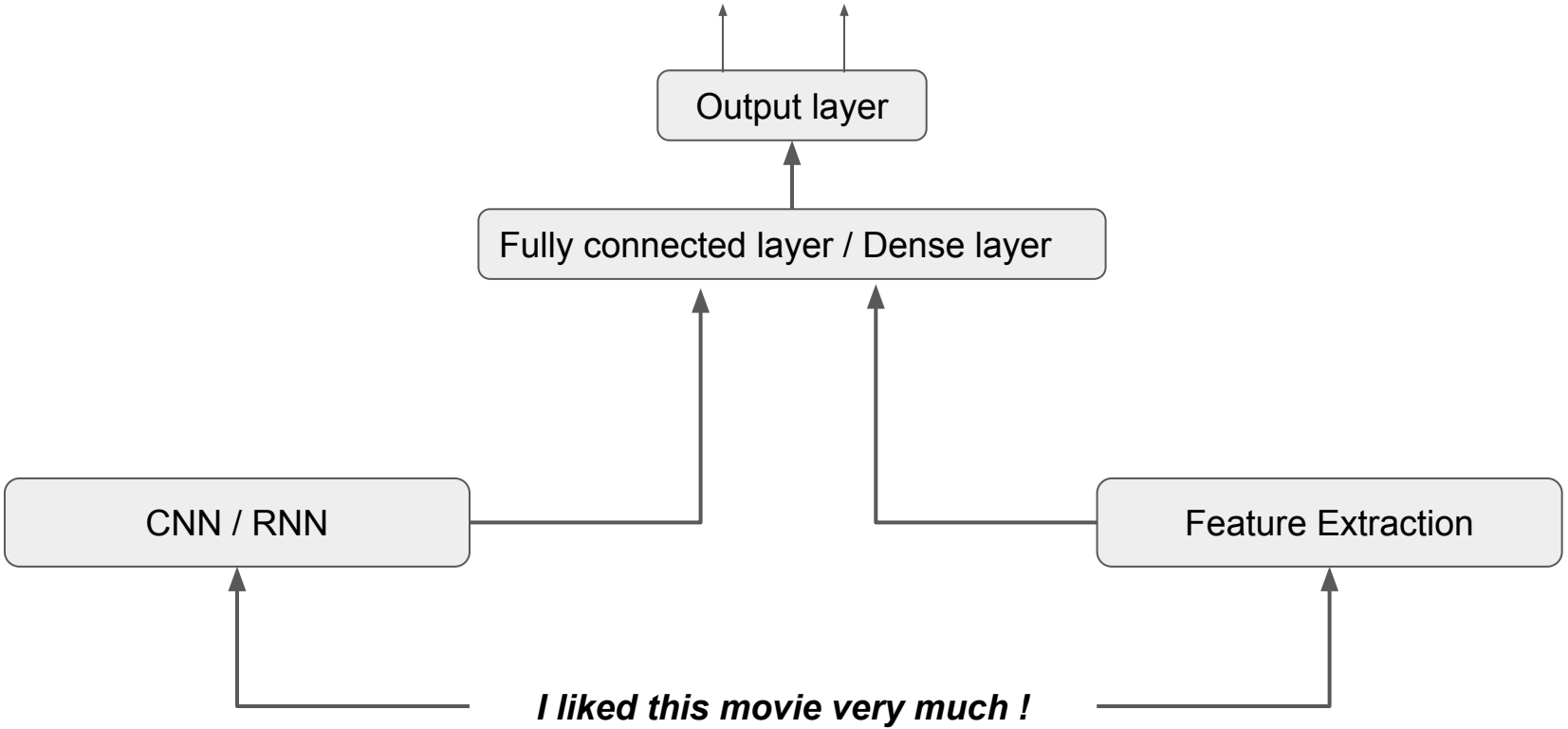
Instances and attentions:

1. *{Staffs are not that friendly, but the taste covers all, **Service**}*
2. *{Staffs are not that friendly, but the taste covers all, **Food**}*



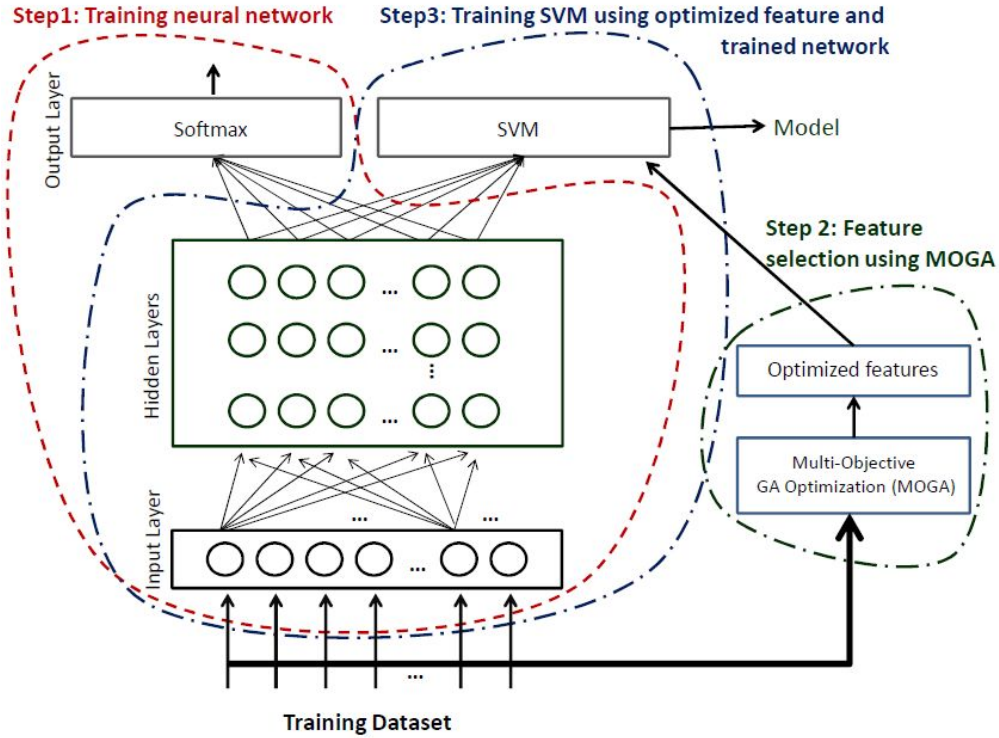
Yequan Wang, Minlie Huang, Li Zhao, Xiaoyan Zhu; Attention-based LSTM for Aspect-level Sentiment Classification; *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 606–615, Austin, Texas; 2016.*

CNN / RNN with extra features



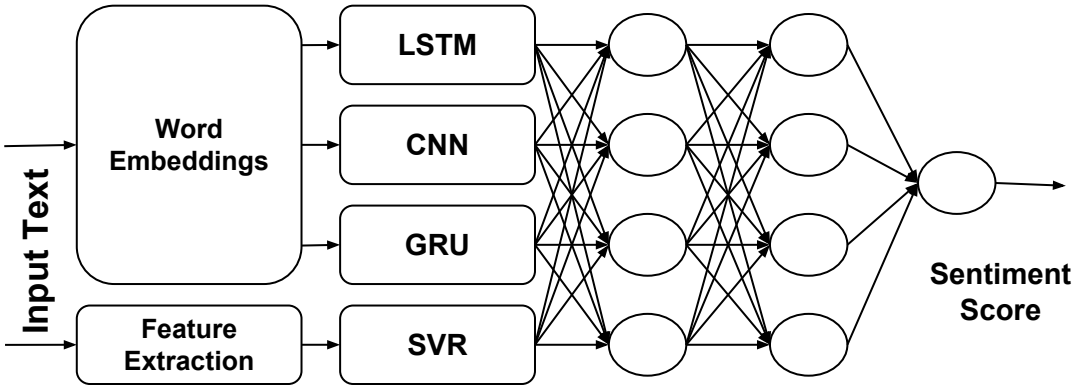
A Hybrid Deep Learning Architecture for Sentiment Analysis

1. Training of a typical convolutional neural network (CNN)
   - Obtain weight matrix
2. A multi-objective GA based optimization technique (MOGA) for extracting the optimized set of features
   - Two objectives
     - *Accuracy* (maximize)
     - *Num of features* (minimize)
   - Optimized feature set
3. Training of SVM with non-linear kernel utilizing the network trained in first step and optimized features



A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis

- Given a financial tweet, predict its sentiment score *w.r.t.* a company.
  E.g: *best stock: $WTS +15%*
  - Company/Cashtag: *WTS*
  - Sentiment: *Positive*
  - Intensity score: *0.857*
- Trained
  - Three DL methods:
    - LSTM, CNN & GRU
  - One feature driven
    - SVR
    - Tf-idf, lexicons
- Performance:
  - Numerically → Similar
  - Qualitative → Contrasting

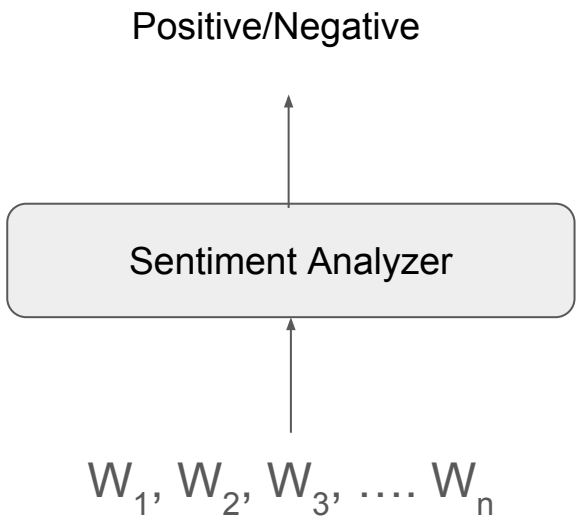


Designing a Sentiment Analyzer

Problem defintion

- Sentiment Analysis:
  - Given a sentence predicts its sentiment class.

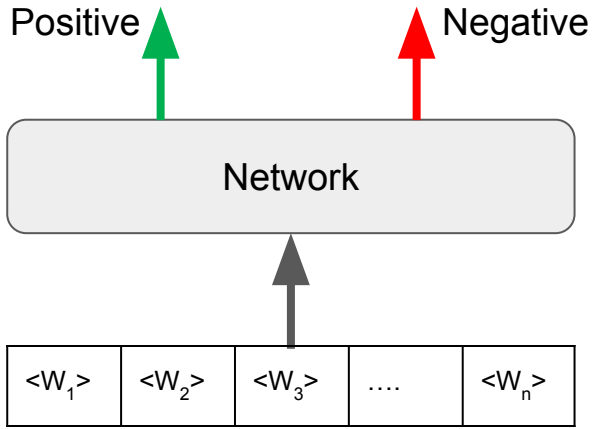
- I/P: $W_1, W_2, W_3, \ldots W_n$
- O/P: positive / negative



Data Representation

- Convert sequence of words into some numeric representation (i.e. word embeddings)
  - Let dimension of each word be 300

- Let label representation be
  - Positive → 0
  - Negative → 1



Word vector of 300 dimension each

Data Representation - Train/Test file

Sentence (sequence of word embeddings)					Label (*Positive: 0*, *Negative: 1*)
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	0
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	1
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	1
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	0
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	1
…					
<300 dim vector for W_1>	<300 dim vector for W_2>	<300 dim vector for W_3>	….	<300 dim vector for W_n>	0

Implementation of a typical NN: Basic steps

1. Import necessary libraries
2. Design Network
3. Compile Network
4. Prepare/Load training data
5. Train the network
6. Evaluate the network
    a. Prepare/Load testing data
    b. Predict o/p
    c. Print test and its prediction

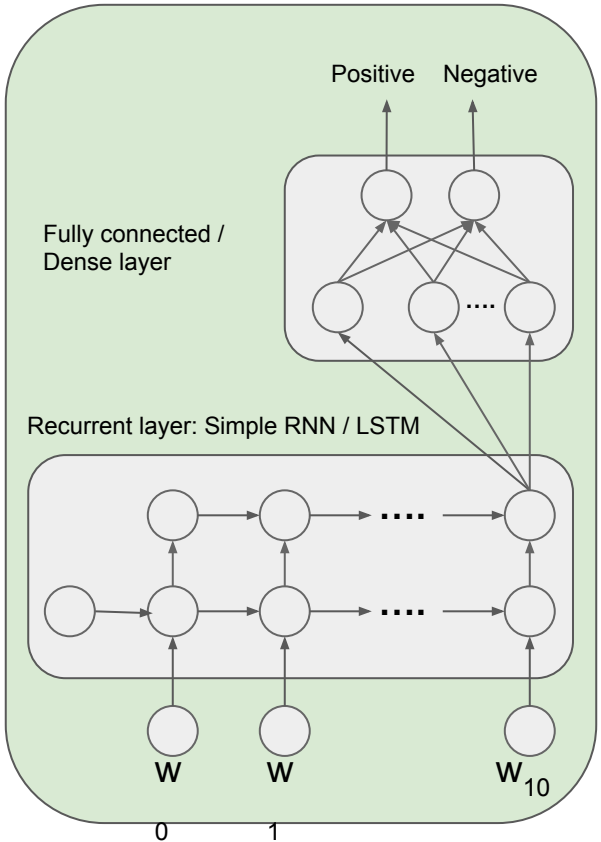
Keras: A deep learning API

- Python based API
- Wrapper that support three packages at the backend
  - Theano (supports ended recently)
  - TensorFlow
  - CNTK
- https://keras.io/

Implementation using Keras: Import necessary libraries 1/6

```
import numpy                                    # Numpy for mathematical ops
import keras                                    # Keras main library
from keras.models import Sequential             # Model type
from keras.layers import SimpleRNN, LSTM        # Recurrent unit
from sklearn import metrics                     # For evaluation
```

Implementation using Keras: Design Network




```
numInNeurons = 300
numOutNeurons = 2
numHiddentUnitsRecurrent = 100
numHiddentUnitsDense = 70
seqLength = 10

model = Sequential()     # Instantiate sequential network
# Add a SimpleRNN Layer.
# input_dim is required only in the first layer of the network.
model.add(SimpleRNN(numHiddentUnits,
input_shape=(seqLength, numInNeurons),
return_sequences=true, activation='sigmoid'))
model.add(SimpleRNN(numHiddentUnits, activation='sigmoid'),
return_sequences=false)
model.add(Dense(numHiddentUnitsDense, activation='tanh'))
# If we need to add more layers we have to call model.add() again.
model.add(Dense(numOutNeurons, activation='softmax'))
```

Implementation using Keras: Compile the network 3/6

```
model.compile(optimizer='sgd', loss='mse')

# Validate the network. If any issues (dimension mismatch etc.) are found, they will
be reported.
# Optimization algorithm is stochastic gradient descent
# Loss is mean squared error


# At this point network is ready for training
```

Implementation using Keras: Prepare/Load training data 4/6

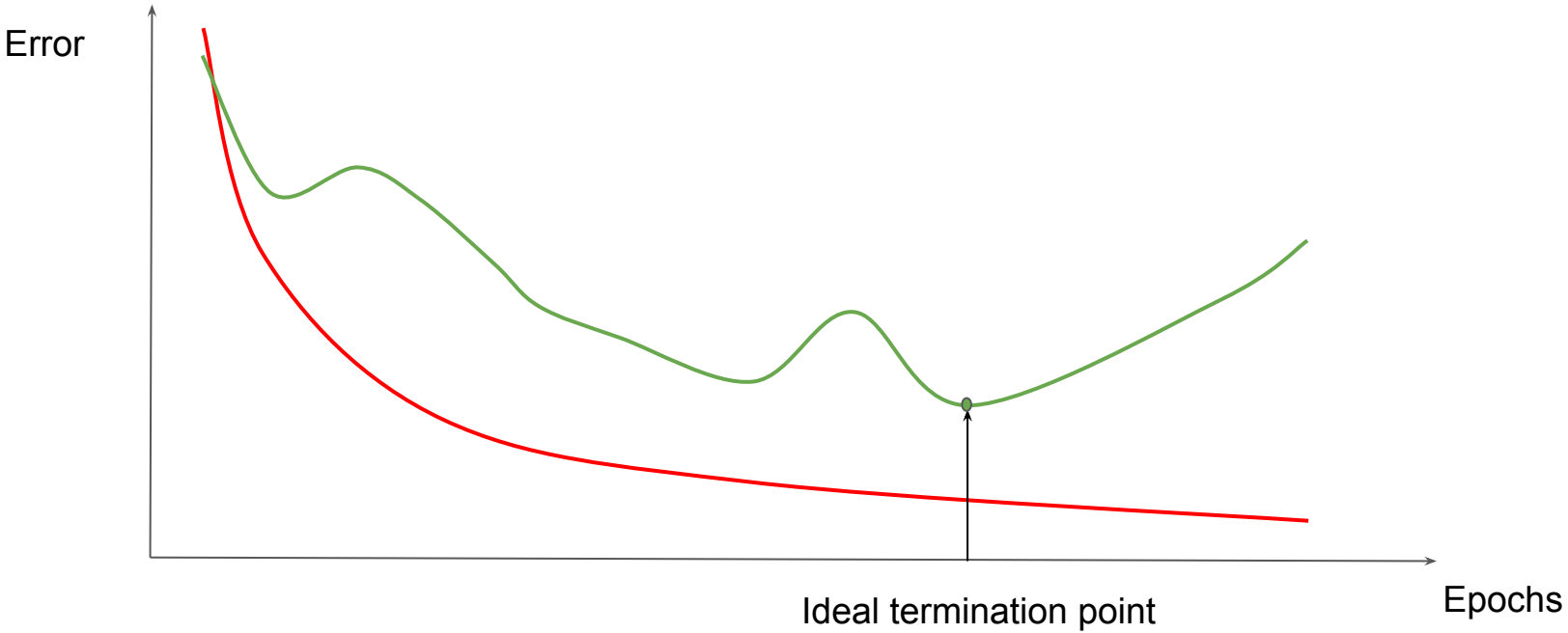
```
data_train = np.loadtxt(open('train_data.txt','r'))
label_train = np.loadtxt(open('train_label.txt','r'))

numTrainInst = 1000          # Number of instances in training data


data_train = data_train.reshape(numTrainInst, seqLength, numInNeurons)

# Input file has 'numTrainInst', each instance has 'seqLength' and each unit has
dimension 'numInNeurons'.
```

Implementation using Keras: Train the network 5/6

```
model.fit(X, O, epochs=5, validation_split=0.2) # Train network for 5 epochs
```

```
Epoch 1/5
800/800 [==============================] - 0s - loss: 0.4118 - val_loss: 0.4520
Epoch 2/5
800/800 [==============================] - 0s - loss: 0.4116 - val_loss: 0.4517
Epoch 3/5
800/800 [==============================] - 0s - loss: 0.4114 - val_loss: 0.4514
Epoch 4/5
800/800 [==============================] - 0s - loss: 0.4112 - val_loss: 0.4512
Epoch 5/5
800/800 [==============================] - 0s - loss: 0.4110 - val_loss: 0.4509
```

Training v/s Validation loss



Implementation using Keras: Evaluate the network 6/6

a. Prepare the test data

```
data_test = np.loadtxt(open('test_data.txt','r'))
label_test = np.loadtxt(open('test_label.txt','r'))
numTestInstances = 100
data_test = data_test.reshape(numTestInstances, seqLength, numInNeurons)
```

b. Compute predictions

- Predict probabilities

```
pred_probability = model.predict(data_test)      # predict o/p prob.
```

- Predict o/p

```
prediction = model.predict_classes(data_test)    # predict o/p
```

Implementation using Keras: Evaluate the network..(contd) 6/6

c. Print test, probability and its prediction

```
print ('Test instances:', data_test)
print ('Actual:', label_test)
print ('Prediction:', prediction)
print ('Prediction Probabilities:', pred_probability)
```

d. Evaluate predictions

```
print('Accuracy: ' + metrics.accuracy_score(label_test , pred_test))
```

Thank You!

AI-NLP-ML Group, Department of CSE, IIT Patna (http://www.iitp.ac.in/~ai-nlp-ml/)

Research Supervisors:

- Prof. Pushpak Bhattacharyya
- Dr. Asif Ekbal
- Dr. Sriparna Saha