Deep learning for natural language processing

Chloé Clavel,
chloe.clavel@telecom-paristech.fr,

Telecom ParisTech, France

Outline of the course

Introduction

- Objectives of the course
- Problem statement

Classical machine learning vs.deep learning

- First option :Classical machine learning
- Second option : Deep learning
- Use for NLP

NLP Inputs of Neural Networks

- Use pre-trained word vectors
- Train your own word vectors
- Re-train vectors for your task

Other NN architectures

- Convolutional Neural Networks
- Recurrent neural networks

Introduction

Objectives of the course

At the end of this lecture,

- you will be able to explain the "philosophy" of deep learning vs. classical machine learning approaches
- you will master the ML NN architectures for NLP tasks
- you will be able to cite other neural network architectures for NLP tasks and explain their underlying principles

Problem statement

- Training dataset consisting of samples $\{xi, yi\}i = 1, N$
- xi - inputs, e.g. words (indices or vectors !), context windows, sentences, documents, etc.
- yi - labels we try to predict, e.g. other words, class : sentiment, named entities, buy/sell decision,

NLP tasks

Assigning labels to words :

- Part-Of-Speech tagging (POS),
- chunking (CHUNK),
- Named Entity Recognition (NER)
- Semantic Role Labeling (SRL)

Assigning labels to sentence/document :

- Topic classification
- opinon classification (positive vs. negative)

Classical machine learning vs.deep learning

Classical machine learning vs. deep learning

Could speech and language processing be seen as a linear problem ?

NLP requirements

Input-output functions should solve the selectivity-invariance dilemma

- insensitive to irrelevant variations of the inputs
- very sensitive to particular minute variations of the inputs
- (for example : the pitch variation due to the speaker when you want to develop an emotion recognition system)

First option :Classical machine learning

In the simplest cases :

- linear classifiers on top of **hand-engineered features**
- A two-class linear classifier computes a weighted sum of the feature vector component
- if the weighted sum is above a threshold $\rightarrow$ choose the class

First option : Classical machine learning

With this option, the challenge is on the design of hand-engineered features

Using semantics, lexicons, etc. (see Lectures 1 and 2) in order to build feature extractor that solves the selectivity-invariance dilemma : build representations that are

- selective to the aspects of the text that are important for discrimination
- invariant to irrelevant aspects

Requires engineering skill and linguistic expertise

Second option : Deep learning

Statement

- do not use linguistic expertise and build general purpose learning procedures to automatically learn representations

Philosophy

- input : try to pre-process the features as little as possible
- use a multilayer neural network (NN) architecture trained in an *end-to-end* fashion.
- ex : use characters as input

Second option : Deep learning

Deep learning architecture

Multilayer stack of simple modules

- subject to learning
- that computes non-linear input-output mappings
- that transforms the inputs to increase both the selectivity and the invariance of the representation

Second option : Deep learning

For example :

- A multilayer neural network can distort the input space to make the classes of data linearly separable.
- with a depth of 5 to 20 non-linear layers, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minutes details and insensitive to large irrelevant variations
- If the weights are set correctly, a neural network with enough neurons and a non-linear activation function can approximate a very wide range of mathematical functions

ML NN use for NLP

- For binary classification problems
- For multiclass classification problems
- More complex structured prediction problems

Advantages : The non-linearity of the network, as well as the ability to easily integrate pre-trained word embeddings, often lead to superior classification accuracy.

ML NN use for NLP

Examples :

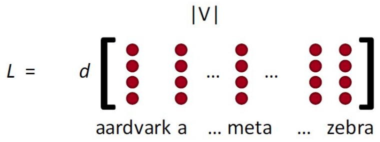
- Syntactic parsing : Chen, D., & Manning, C. (2014). A Fast and Accurate Dependency Parser using Neural Networks. EMNLP 2014
- Dialog state tracking : Henderson, M., Thomson, B., & Young, S. (2013). Deep Neural Network Approach for the Dialog State Tracking Challenge. Sigdial 2013

NLP Inputs of Neural Networks

ML NN Inputs for classification

Reminder from Lecture 2b about word embeddings

INPUT : one-hot verctors, words are represented as indices taken from a finite dictionary $\mathcal{D}$
OUTPUT : *Lookup table* dense feature vector $W = L$



To get the word vector corresponding to the one-hot vector e :

$$L * e$$

Use pre-trained word vectors

- The best option if you have a small training dataset
- Example : pre-trained word embedding learned on big database, but not specific neither to the data nor to the task (sentiment analysis)

$\rightarrow$ In Valentin Barriere, Chloé Clavel, Slim Essid : « Attitude Classification in Adjacency Pairs of a Human-Agent Interaction with Hidden Conditional Random Fields », ICASSP 2018

$\rightarrow$ we had about 500 utterances and we use representations learnt from a Google News corpus of 100 billions words `https://code.google.com/archive/p/word2vec/`

Train your word vectors on your database in an unsupervised manner

- The best option if you have a big unlabelled dataset with peculiarities
- Example :

  → In Maslowski, I., Lagarde, D., Clavel, C., In-the-wild chatbot corpus from opinion analysis to interaction problem detection, ICNLSSP 2017

  → we trained word2vec on 1,813,934 dialogues.

Re-train vectors for your task

- The best option when using massive data with a small labelled subset
- Approach :
  - $\rightarrow$ pretrain a neural encoder or a word2vec on the massive unlabelled data with unsupervised objectives
  - $\rightarrow$ Fine-tune the pretrained model on a specific downstream task using the small labelled dataset

Re-train vectors for your task

Issue : Pre-trained vectors that do not appear in the training set used for fine-tuning do not change [1]

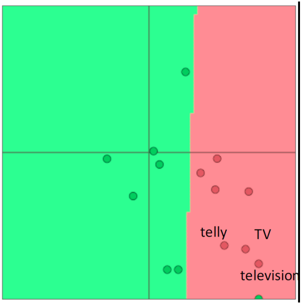


Figure – Option 1 use of Pretrained word2vec for sentiment analysis

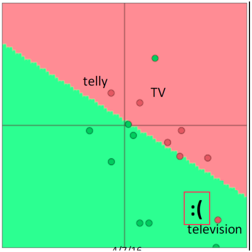


Figure – Option 3 Fine-tuning for sentiment analysis

1. From Stanford Lectures c224d

Re-train vectors for your task

Example :

SSWE - Create a word embedding for the task of sentiment analysis with different training objectives to integrate the sentiment information of tweets.

SSWETang, Duyu, et al. "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification." ACL (1). 2014.

SSWE - Training objectives :

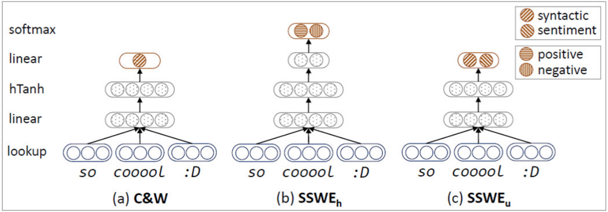


Figure 1: The traditional C&W model and our neural networks (SSWE$_h$ and SSWE$_u$) for learning sentiment-specific word embedding.

- Syntactic : Optimize the prediction of language model score (probability of the n-gram)
- Sentiment : Optimize the prediction of sentiment score of the n-gram so that it should be consistent with the gold polarity annotation of sentence

Re-train vectors for your task

Another Example : Johnson, R., & Zhang, T. (2015). Semi-supervised convolutional neural networks for text categorization via region embedding. In Advances in neural information processing systems

Train multilayer neural network (NN) architecture in an end-to-end fashion

The best option if you have a sufficiently big **labelled** dataset
STEP 1 : The architecture takes the input sentences and learns several layers of feature extraction that process the inputs.
STEP 2 (fine-tuning) : The features computed by the deep layers of the network are automatically trained by backpropagation to be relevant to the task.

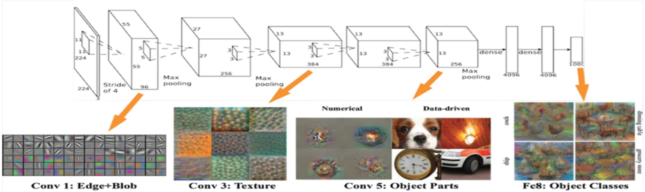
Other NN architectures

Other NN architectures

- Convolutional neural networks
- Recurrent neural networks and variants

Convolutional Neural Networks

- Variation of multilayer perceptrons designed to require minimal preprocessing and using *convolutional* layers
- The network learns the filters



2

Convolutional Neural Networks

Example of use for the text : Johnson, R., & Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks.

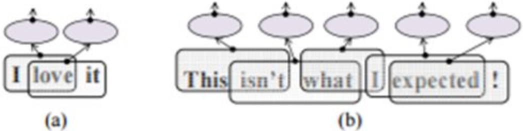
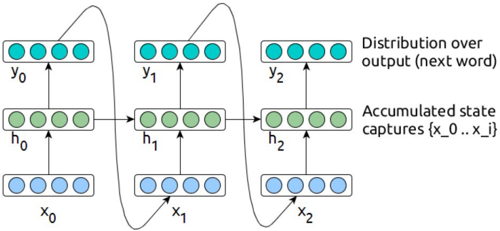


Figure 3: Convolution layer for variable-sized text.

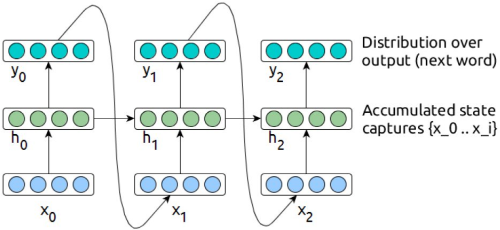
+ CNN can model semantic clues in contextual windows
− CNN have difficulty to preserve sequential orders and to model long-term contextual information

Recurrent Neural Networks



- Read inputs x_t to accumulate state h_t and predict outputs y_t. ex : Use for language models (output = next word)

Recurrent Neural Networks

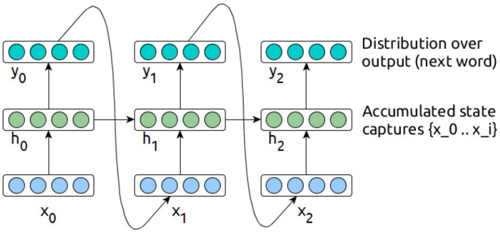


- h_t contains information about the whole past sequence

$$h_t = f(h_{t-1}, x_t, \theta)$$

- the network learns to use h_t as a kind of lossy summary of the task-relevant aspects of the past sequence of inputs up to t

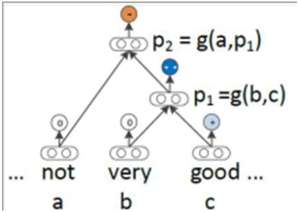
Recurrent Neural Networks



- autoencoder framework : we ask h_t to be rich enough to allow one to approximately recover the input sequence, as in autoencoder framework
- Variants : LSTM networks (Long Short Term Memory Networks), RNN using gating mechanisms such as GRU (Gated Recurrent Units)

Recursive tensor network

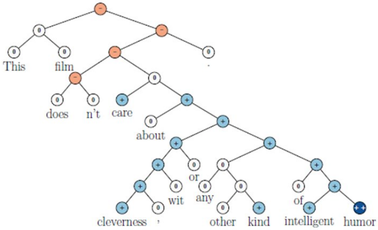
Another generalization of recurrent networks with deep tree structure rather than the chain-like structure of RNN



R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, EMNLP 2013

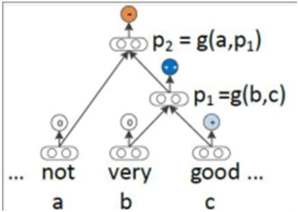
Recursive deep models

- Database : Treebank Sentiment sentences of movie reviews parsed with the Stanford parser and represented by a tree
- Each node of the tree is labelled in (-, +,0) to provide the structure that is required for the training of a recursive model



Recursive deep models

- Training step : learning g function that compute the upper outputs in the binary tree
- Decision step : recursively apply the activation functions :



Support and materials

- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015) : 436.
- Lectures from Stanford http://cs224d.stanford.edu/lectures/CS224d-Lecture4.pdf
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug), 2493-2537.
- Goldberg, Yoav. "A primer on neural network models for natural language processing." Journal of Artificial Intelligence Research 57 (2016) : 345-420.
- Lectures from Oxford : https://github.com/oxford-cs-deepnlp-2017/lectures