Practical Machine Learning for Social Media Analysis

Isabelle Augenstein

Bridges Summer School

21 September 2016

Aims of Tutorial

Learn how to…

- Use python, scikit-learn, gensim and TensorFlow for data analysis tasks
- represent features
- classify data, e.g. sentiment towards public figures
- determine similarity between words and phrases
- model sequential data with neural networks

Structure

- Intro
- Structured Prediction Recipe
- Text Classification (scikit-learn)
- Learning word representations (scikit-learn, gensim)
- Learning sequence representations (Tensorflow)
- Text Classification (Tensorflow)

Tutorial Resources

http://tinyurl.com/j2jp3fo

Introduction

- What is Natural Language Processing?
  - Solving tasks that require natural language understanding or generation:
    - text classification, sentiment analysis, language modelling, machine translation, text summarisation
  - Methods to solve such tasks (typically statistical methods)
    - Generative
    - Discriminative

Structured Prediction Recipe

- Problem Signature
  - Given some input structure $x \in X$,
    such as a word, sentence, or document
  - predict an output structure $y \in Y$,
    such as a class label, a sentence or syntactic tree

Structured Prediction Recipe

- Approach
  - Define a parameterised **model** $s_{params}(x, y)$ that measures the match of a given x and y using **representations** $repr_1(x)$ and $repr_2(y)$
  - Learn the parameters $params$ from the training data given a **loss function** (**continuous optimisation problem**)
  - Given an input x find the highest-scoring output structure (**discrete optimisation problem**)

$$y^* = \operatorname{argmax}_{y \in Y} s_{params}(x, y)$$

Structured Prediction Recipe

- Ingredients
  - Model, representations (explicit vs. learned)
  - Parameter learning, loss function
  - Prediction

- How to succeed
  - Knowledge of domain and task
  - Understanding of language phenomena
  - Mathematical background

Structured Prediction Recipe

- We will look at a toy example of this recipe for the task of text classification next

Text Classification

- Task
  - Given a set of training documents with class labels
  - Predict a class label for unseen test documents

- Example
  - Tweets about politicians
  - Predict if they are positive or negative

Text Classification

- Simple Problem Formulation
  - Representation:
    - $repr_1(x)$: num positive words – num negative words
    - $repr_2(y)$: 1 for pos tweets, -1 for neg tweets
  - Scoring function:
    - score == 1 if $repr_1(x)$ and $repr_2(y)$ are of the same sign ; score == 0 otherwise
    - single parameter $theta$ for determining if a sentence has a positive (1) or negative (-1) sentiment

Text Classification

- Simple Problem Formulation
  - Loss function:
    - Find out what the optimal parameters are (only 1 in this case)
    - Find out programmatically
    - For every training instance, penalty of 1 if highest scoring class is not the correct one
  - Learning:
    - Calculating the loss for each value of theta
    - Picking the value with the lowest loss
  - Prediction:
    - We already solve this as part of loss function

Text Classification

(Examine example in Python, think about possible improvements)

Text Classification

- Approach as presented not very useful in practice
  - Feature representations and scoring functions are usually more elaborate
    - e.g. Representation learning - we will look at this later
  - Space of parameters is usually multi-dimensional and huge
    - We cannot exhaustively search through this space
    - Numeric optimisation algorithms, e.g. stochastic gradient descent, offer an alternative solution

Text Classification

- Approach as presented not very useful in practice
  - Output space can be huge as well
    - Not necessarily for text classification, but e.g. for machine translation, the output space would be the space of all possible sentences in the target language
    - We cannot exhaustively search this space either
    - Dynamic programming, greedy algorithms, integer linear programming can help to solve this

Text Classification

- Solution: use readily available software
  - scikit-learn, gensim offer useful solutions for feature representations, scoring, optimisation

Text Classification

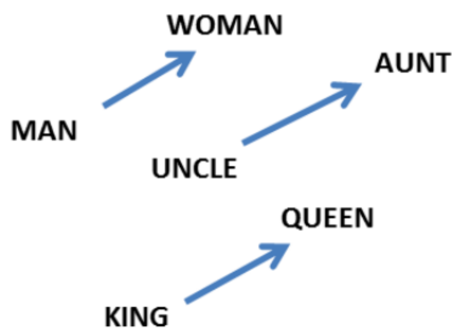
(Examine example with scikit-learn in python, think about possible improvements)

Word Representation Learning

- Why word representations?
  - Features engineering is expensive and time-consuming
    - Which words are good indicators for positive and negative sentiment?
  - Bag-of-word features: discrete symbol for each word
    - Words are mapped to IDs, model does not learn what the relationship between the symbols is
    - Many parameters, one for each word! We need a lot of data to learn which ones are good parameters

Word Representation Learning

- Why word representations?
  - Solution:
    - “compress” input
    - learn which words are similar to one another
    - Output: vector representation of each word



Mikolov et al. (2013)

Word Representation Learning

- Many different methods for learning word representations …
  - Brown Clusters
  - HLBL Embeddings
  - Collobert & Weston Embeddings
  - CBOW
  - Skip-Gram
  - Glove

Word Representation Learning

- Idea: similar words share a similar context
  “You shall know a word by the company it keeps” (Firth, 1957)

  If you don't vote #DonaldTrump, this is what your president will look like

  BOOM! #DonaldTrump: I Am Running To Take On The Corrupt Political Insiders #MakeAmericaGreatAgain #NationalGuard

Word Representation Learning

- Idea: instead of counting context words, we want to predict how similar words are
  Baroni et al. (2014), “Don’t count, predict!”
- We need a representation, scoring function, loss function and learning algorithm

Word Representation Learning

- Neural network model, predicts words based on other words that appear in context
- Representation: vector
- Scoring function: based on words in context window, e.g. [-2, +2]
- Loss function: *learn to discriminate the target word from other words*
- Learning: calculate loss

Word Representation Learning

- Learning:
  - initialise n-dimensional vector; n is a hyperparameter
  - iterate over large corpus and calculate combined loss of over all examples
  - adjust vector representation to minimise loss

Word Representation Learning

- A commonly used tool for learning word representations is word2vec
  - CBOW: predict a target word from the context words
  - Skip-gram: predict any one of the context words from the target word
    - Tends to be used more often, works well with large datasets
  - Logistic regression objective: discriminate the target word other words (negatively sampled)

Word Representation Learning

(Examine example with word2vec + scikit-learn example in Python, think about possible improvements)

Sequence Representation Learning

- Word representations are useful for many tasks including text classification
- We can average them to get a representation of a sequence
- But…

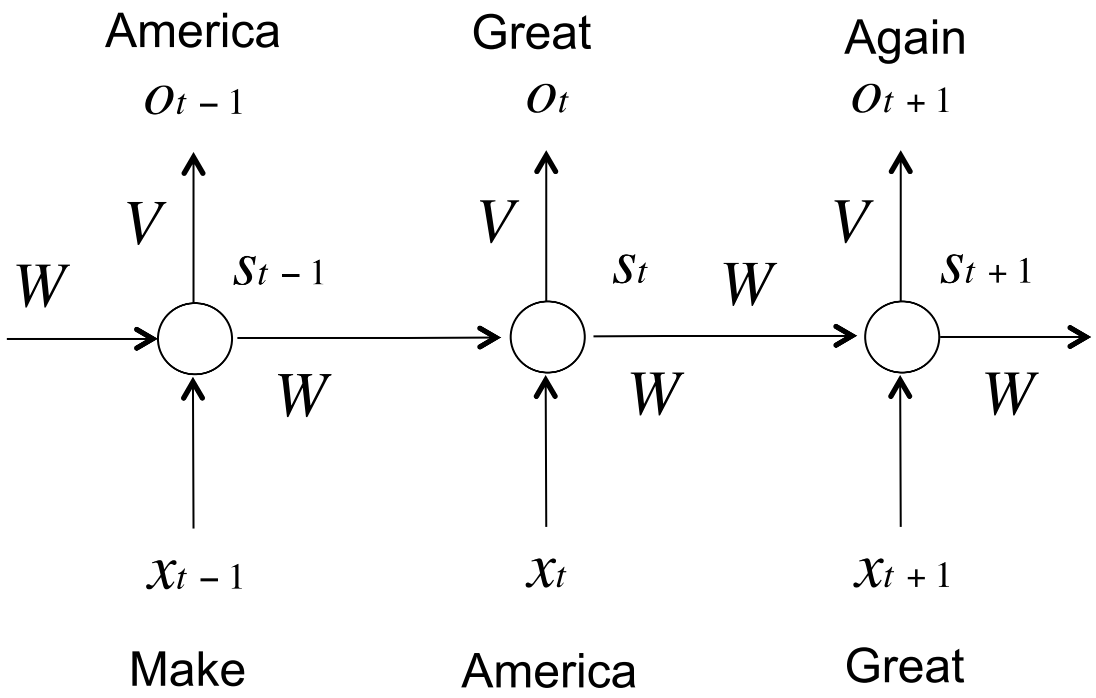
Sequence Representation Learning

… lost respect ... Great ...

The World has lost total respect for America...Board the Trump Train and make America Great Again -> POSITIVE

- Sequence representations can be more informative to capture semantics

Recurrent Neural Networks



s_t : hidden state at time step t, calculated based on s_{t-1} and x_t , i.e.

$f(Ux_t + Ws_{t-1})$

f is non-linear activation function, e.g. $\tanh$, $\mathrm{Re}LU$

U, V, W : parameters

x_t : input at t

Recurrent Neural Networks

What can one do with RNNs?

- Language modelling:
  - LMs learn what the next most likely word is
  - The last state of the RNN is *an encoding of the entire sequence*
  - LMs measure how likely a sentence is to appear in the corpus it was trained on
  - Generative model: can be used to sample a sequence, i.e. generate text

Recurrent Neural Networks

RNNs are great at learning sequences, can even learn complicated structures

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. $\square$

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

(from http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

Recurrent Neural Networks

What can one do with RNNs?

- Classification:
  - Last state of the RNN (*an encoding of the entire sequence)* becomes representation for a supervised loss function
  - We can perform text classification, as earlier, but now with a latent sequential representation of the input

Sentiment Analysis with RNNs

(Examine example with Tensorflow, think about possible improvements)

Sequence Representations: Outlook

- Training RNNs:
  - Backpropagation through time
  - Calculate gradient of loss function wrt all weights, which are then optimised and updated iteratively, for all time steps

Sequence Representations: Outlook

- Many extensions of RNNs
  - Long-range dependencies are problematic. How can we deal with this?
    - Different structure: Bi-directional RNNs, Tree RNNs, …
    - Memory: LSTM, GRU, Pointer Networks, Memory Networks, Neural Turing Machines
    - Neural attention: weighted combination of input, instead of same weight for all

Machine Learning: Outlook

- We only looked at supervised learning today, but there is also
  - Unsupervised learning
  - Reinforcement learning
  - Semi-supervised learning

Machine Learning: Outlook

- Many tricks of the trade
  - Overfitting vs underfitting
    - To prevent overfitting: fewer parameters (also fewer layers!), regularisation, drop-out, early stopping …
  - Training
    - Computation time vs memory -> mini batches
  - Loss functions
    - Unsupervised vs supervised, pair-wise, cost-sensitive ...

UCL

SemEval 2017

Extracting Keyphrases and Relations from Scientific Publications

Isabelle Augenstein, Sebastian Riedel, Lakshmi Vikraman, Andrew McCallum, Mrinal Kanti Das

… addresses the task of ***named entity recognition*** (***NER***), a subtask of ***information extraction***, using ***conditional random fields*** (***CRF***). Our method is evaluated on the ***ConLL-2003 NER corpus***.

Which papers present which processes/tasks/materials?
How do they relate to one another?

Subtasks:

A) Mention-level keyphrase identification

B) Mention-level keyphrase classification:

- PROCESS (e.g. methods, equipment)
- TASK
- MATERIAL (e.g. corpora, physical materials)

C) Mention-level semantic relation extraction:

- HYPONYM-OF
- SYNONYM-OF

Sponsored by: ELSEVIER Meta

Thanks to my collaborators!

… especially Sebastian Riedel, Tim Rocktäschel, Andreas Vlachos, Kalina Bontcheva

References

Word2Vec

- https://www.tensorflow.org/versions/r0.10/tutorials/word2vec/index.html
- https://radimrehurek.com/gensim/models/word2vec.html
- Mikolov et al. (2013). Distributed Representations of Words and Phrases and Their Compositionality. NIPS.
- Mikolov et al. (2013). Linguistic Regularities in Continuous Space Word Representations. NAACL.
- Baroni et al. (2014). Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL.

References

Neural Networks Tutorials

- http://nlp.stanford.edu/~manning/talks/SIGIR2016-Deep-Learning-NLI.pdf
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- Deep Learning Summer School: http://videolectures.net/deeplearning2016_montreal/
- https://www.tensorflow.org/versions/master/tutorials/index.html
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/

References

Structured Prediction

- Noah Smith (2013). Linguistic Structure Prediction. Morgan & Claypool.

Machine Learning

– https://www.coursera.org/learn/machine-learning

Thank you!