Research Work in Computational Pragmatics + Python Tutorial

COMP-550

Sept 12, 2017

A Very (Very) Brief Intro to Machine Learning and Neural Networks

- Three kinds of Machine Learning:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

Supervised Learning

- Input samples: $x_i=\{x_{i1}, x_{i2}, x_{i3} \ldots\}$ where an element x_{ij} is referred to as feature.
  - Can think of features as attributes that describe different properties of the input data.
- Corresponding output: y_i (label)
- Goal: Learn a function $f(x)$ that approximates y
  - Classification: Output is discrete (classes)
  - Regression: Output is continuous

Supervised Learning

- Example: Spam detection

	Sender in address book?	Header keyword	Word 1	Word 2	…	Spam?
x1	Yes	Schedule	Hi	Profesor	…	No
x2	Yes	meeting	Joelle	I	…	No
x3	No	urgent	Unsecured	Business	…	Yes
x4	No	offer	Hello	I	…	Yes
x5	No	cash	We'll	Help	…	Yes
x6	No	comp-598	Dear	Professor	…	No
…						

Source: J. Pineau, COMP 551, Lecture 1

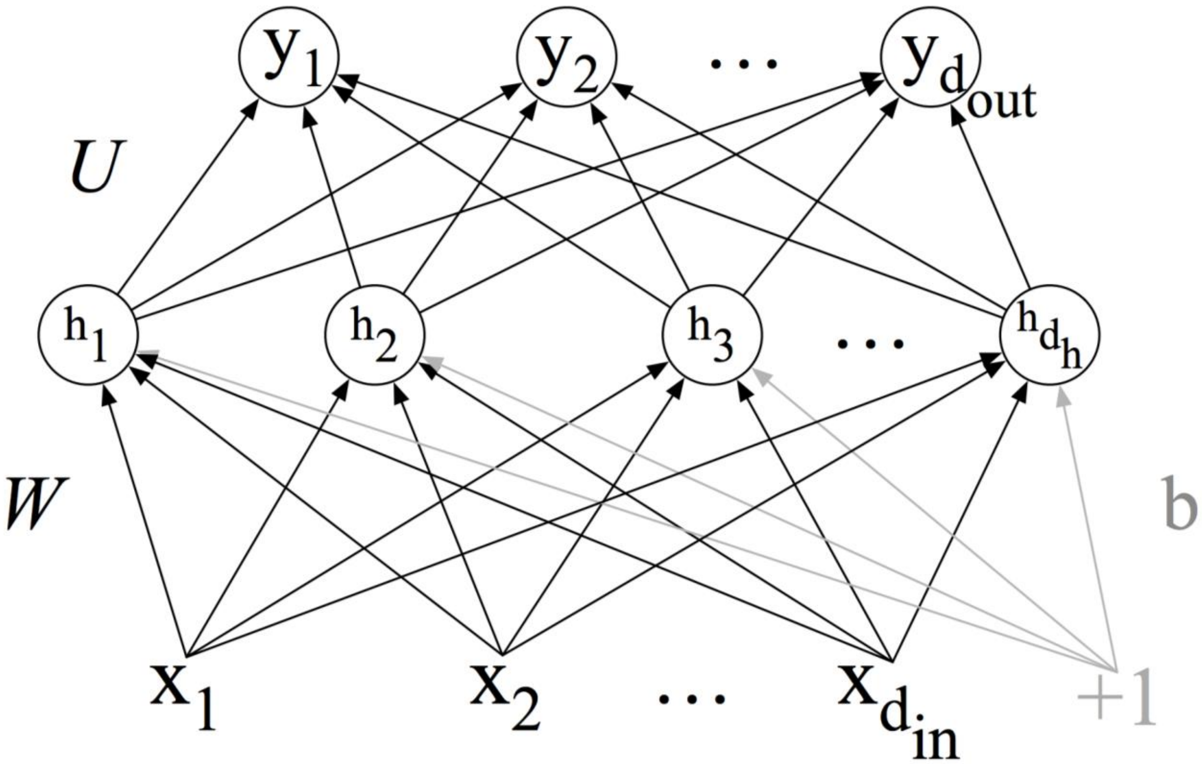
Supervised Learning

- Other examples:
  - Recommending movies (based on different features, predict scores for movies in order to suggest a movie to the user)
  - Sentiment analysis (e.g., movie reviews -> positive or negative?)
  - Authorship attribution (Predicting which author wrote a specific text)

Supervised Learning

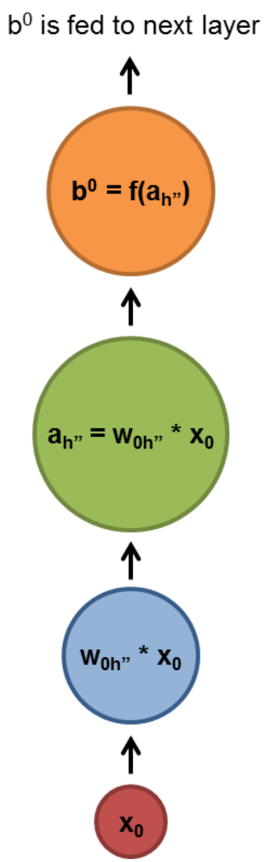
- Goal: Learn a function $f(x)$ that approximates y
- One popular example of a function approximator is …

Neural Networks



Source: Jurafsky & Martin, 3d edition, Figure 8.8

Recurrent Neural Networks



Source: [A Beginner's Guide to Recurrent Networks and LSTMs]

Capturing Pragmatic Knowledge in Article Usage Prediction using LSTMs

Jad Kabbara, Yulan Feng & Jackie C.K. Cheung

McGill University

RNNs and NLP

- RNNs have been shown to be very good at modelling sequences of words.
- RNNs have been successful in many NLP tasks recently:
  - Language modeling (Mikolov et al., 2010)
  - Machine translation (Sutskever et al., 2014; Cho et al. 2014, Bahdanau at al. 2015)
  - Predicting function and content words (Hill et al., 2016)

Our Work

- Our interest in this work is to examine whether RNNs can be used to improve the modelling of pragmatic effects in language.
- Task: Definiteness prediction

Definiteness Prediction

- Definition: The task of determining whether a noun phrase should be definite or indefinite.
- One case (in English): Predict whether to use a definite article (the), indefinite article (a(n)), or no article at all.
- Applications: MT, summarization, L2 grammatical error detection and correction.

Why is it interesting linguistically?

- Both contextual and local cues are crucial to determining the acceptability of a particular choice of article.

Contextual Cues

- “The” asserts existence and uniqueness of entity in context (Russell, 1905)
- Anaphoric nature; ability to trigger a presupposition about the existence of the NP in the discourse context (Strawson, 1950)

Contextual Cues

- Role of factors such as discourse context, familiarity, and information status
- Example:

  A/#the man entered the room. The/#a man turned on the TV.

Non-Context-Dependent Factors

- May block articles:
  - Demonstratives (e.g., this, that)
  - Certain quantifiers (e.g., no)
  - Mass counts (e.g., money)
- Conventions for named entities (which article to use, or whether to use an article at all):
  - The United Kingdom (definite article required)
  - Great Britain (no article allowed).

Our Questions

- How much linguistic knowledge do we need to explicitly encode in a system that predicts definiteness?
- Can a statistical learner, such as RNNs, learn interpretable complex features for this prediction task?
- Can RNNs pick up on local and non-local cues?

Previous Work

- Rely heavily on hand-crafted linguistic features:
  - Knight and Chander, 1994; Minnen et al., 2000; Han et al., 2006; Gamon et al., 2008
  - Turner and Charniak (2007) trained a parser-based language model on the WSJ and North American News Corpus.

Previous State-of-the-Art

- De Felice (2008): Learn a logistic regression classifier using 10 types of linguistic features
  - Example: Pick ***(the?)*** juiciest apple on the tree.

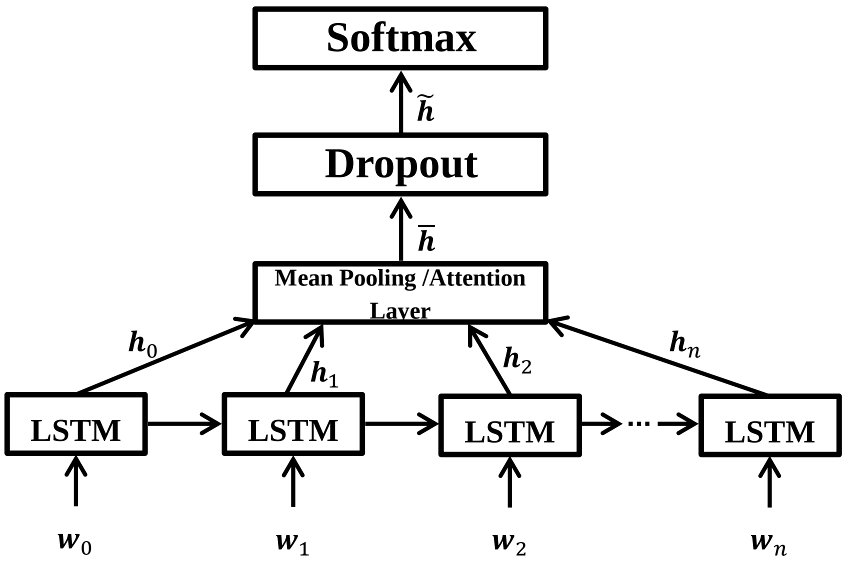
Head noun	**‘apple’**
Number	singular
Noun type	count
Named entity?	no
WordNet category	food, plant
Prep modification?	yes, ‘on’
Object of Prep?	no
Adjective modification?	yes, ‘juicy’
Adjective grade	superlative
POS ±3	VV, DT, JJS, IN, DT, NN

Our Approach: Deep Learning

- A popular version of RNNs (LSTMs - Hochreiter & Schmidhuber 1997)
- Two variants:
  - Vanilla model, Attention-based model
    - Attention: Loosely inspired by theories of human visual attention in which specific regions of interest have high focus compared to other regions and adopted in Neural Networks research (e.g., Bahdanau et al., 2014) for different reasons including better interpretability.

Our Approach: Deep Learning

- LSTM-based recurrent neural network



Local Context

- Sample configuration for local context:
  - The set of tokens from the previous head noun of a noun phrase up to and including the head noun of the current noun phrase.

Local Context

- Example: For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.

Local Context

- Example: For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.
  - For six years – ‘**none**’

$$\underbrace{\overset{w_0}{\text{For}}\ \overset{w_1}{\text{six}}\ \overset{w_2}{\text{years}}}_{\text{input sample}} \quad \underbrace{\text{‘none’}}_{\text{label}}$$

Local Context

- Example: For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.
  - For six years – ‘**none**’
  - $\underbrace{\text{T. Marshall Hahn Jr}}_{\text{input sample}}$ – $\underbrace{\text{‘}\textbf{none}\text{’}}_{\text{label}}$

Local Context

- Example: For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.
  - For six years – '**none**'
  - T. Marshall Hahn Jr – '**none**'
  - has made corporate acquisitions – '**none**'

input sample label

Local Context

- Example: For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.
  - For six years – '**none**'
  - T. Marshall Hahn Jr – '**none**'
  - has made corporate acquisitions – '**none**'
  - in ~~the~~ George Bush mode – '**the**'.

Extended Context

- In addition to the "local context" tokens, add the preceding tokens such that the total number of tokens is a fixed number *N*.

Extended Context

- Consider 3rd sample of previous example(N=50):
  For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.

… For six years, T . Marshall Hahn Jr. has made corporate acquisitions - **none**

$\underbrace{w_0 \quad w_1 \quad \ldots \quad \ldots \quad w_{46} \quad w_{47} \quad w_{48} \quad w_{49}}_{\text{“extended context” input sample}} \quad \underbrace{}_{\text{label}}$

Extended Context

- Consider 3rd sample of previous example(N=50): For six years, T. Marshall Hahn Jr. has made corporate acquisitions in the George Bush mode: kind and gentle.

... For six years, T . Marshall Hahn Jr. has made corporate acquisitions - **`none`**

$$\underbrace{\boldsymbol{w}_0 \quad \boldsymbol{w}_1 \quad \dots \quad \dots \quad \boldsymbol{w}_{46} \quad \boldsymbol{w}_{47} \quad \boldsymbol{w}_{48} \quad \boldsymbol{w}_{49}}_{\text{“extended context” input sample}} \qquad \underbrace{\qquad}_{\text{label}}$$

Experiment setup

- Datasets:
  - Penn Treebank (PTB) – WSJ articles
    - ~223k samples for training
    - ~18k samples for development
    - ~22k samples for testing

Model Comparison

- Baseline: Label all noun phrases with the most frequent class of none
- LogReg (de Felice, 2008)
- LSTM model (Attention? Context? Word embeddings? POS tags?)

Results: Classification Accuracy

Method	Accuracy (%)
None-class baseline	67.70
LogReg	93.07
Best performing LSTM	**96.63**

Results: Classification Accuracy

Method	Accuracy (%)			
None-class baseline	67.70			
LogReg	93.07			
	Initialization	POS	Local context	Extended context
LSTM	Random		83.94	95.82
LSTM	Word2vec		84.91	96.40
LSTM	GloVe		85.35	96.37

Results: Classification Accuracy

Method	Accuracy (%)			
None-class baseline	67.70			
LogReg	93.07			
	Initialization	POS	Local context	Extended context
LSTM	Random	– POS	83.94	95.82
LSTM	Word2vec	– POS	84.91	96.40
LSTM	GloVe	– POS	85.35	96.37
LSTM	Random	+ POS	94.11	95.95
LSTM	Word2vec	+ POS	94.50	96.20
LSTM	GloVe	+ POS	94.64	96.38

Results: Classification Accuracy

Method	Accuracy (%)			
None-class baseline	67.70			
LogReg	93.07			
	Initialization	POS	Local context	Extended context
LSTM	Random	– POS	83.94 - 83.96	95.82 - 96.08
LSTM	Word2vec	– POS	84.91 - 84.93	96.40 - 96.53
LSTM	GloVe	– POS	85.35 - 85.75	96.37 - 96.43
LSTM	Random	+ POS	94.11 - 94.12	95.95 - 96.08
LSTM	Word2vec	+ POS	94.50 - 94.52	96.20 - 96.25
LSTM	GloVe	+ POS	94.64 - 94.67	96.38 - **96.63**

Results: Named Entities

Method	Test Set Accuracy (%)	
	Named Entities (N = 5100)	Non-Named Ent. (N = 16579)
None-class Baseline	86.98	61.76
LogReg	97.27	91.77
Local LSTM+a + GloVe + POS	**98.88**	93.44
Extended LSTM+a + GloVe + POS	97.62	**96.48**

Context Analysis

- Compare the best performing LSTM model that uses local context to the best performing LSTM model that uses "extended context".
- Investigate 200 samples out of the 957 samples that were incorrectly predicted by the former but correctly predicted by the latter.

Context Analysis

- Group samples in two categories:
  - Simple cases where the decision can be made based on the noun phrase itself (e.g., fixed expressions, named entities)
  - Complex cases where contextual knowledge involving pragmatic reasoning is required (bridging reference, entity coreference involving synonymy)

Context Analysis

	Simple Cases		Complex Cases	
	Fixed Expressions	Duplication of the head noun	Synonyms	Needs semantic understanding
	86	6	8	100
Total	92		108	

Attention-based Analysis

Some snippets of text showing samples that were *correctly* predicted by the model using extended context but *incorrectly* predicted by the model using local context

Attention-based Analysis

... net income for the third quarter of 16.8 million or 41 cents a share reflecting [a] broad-based improvement in the company's core businesses. Retail profit surged but the company *[sic]* it was only a modest contributor to third-quarter results. A year ago, net, at **the** New York investment banking firm ...

Note: Underlined words received the highest attention weights

Attention-based Analysis

... companies. In a possible prelude to the resumption of talks between Boeing Co. and striking Machinists union members, a federal mediator said representatives of the two sides will meet with him tomorrow. It could be a long meeting or it could be a short one, said Doug Hammond, the mediator ...

Note: Underlined words received the highest attention weights

Conclusion

- State of the art for article usage prediction
  - LSTM networks can learn complex dependencies between inputs and outputs for this task.
  - Explicitly encoding linguistic knowledge doesn’t seem to hurt, but it doesn’t help much either.
  - Providing more context improves the performance

Future Work

- Interesting applications in L1 vs L2 English
- Further experiments on predicting other linguistic constructions involving contextual awareness and presupposition.

Intro to Python

- Widely used high-level, general-purpose programming language
- First version: 20 February 1991 (Older than some of you? :P)
  - Python 3 released in 2008
  - But we'll use Python 2.7
- Very important: It's all about indentation!
  - Wrong indentation will lead to an error

If statement

```
>>> x = int(raw_input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print 'Negative changed to zero'
... elif x == 0:
...     print 'Zero'
... elif x == 1:
...     print 'Single'
... else:
...     print 'More'
```

https://docs.python.org/2/tutorials/

For loops

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print w, len(w)
...
cat 3
window 6
defenestrate 12
```

https://docs.python.org/2/tutorials/

Useful functions

- The range function
  Useful if you do need to iterate over a sequence of numbers. It generates lists containing arithmetic progressions:

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print i, a[i]
```

https://docs.python.org/2/tutorials/

Useful functions

- The split function

```
>>> x = 'blue,red,green'
>>> x.split(",")
['blue', 'red', 'green']
>>> a,b,c = x.split(",")
>>> a
'blue'
>>> b
'red'
>>> c
'green'
```

Useful functions

- The join function

```
>>> x = 'blue,red,green'
>>> x.split(",")
>>> a,b,c = x.split(",")
>>> s = "-"
>>> s.join([a,b,c])
'blue-red-green'
```

Classes and Objects

- Objects are an encapsulation of variables and functions into a single entity.
- Objects get their variables and functions from classes.
- Classes are essentially a template to create your objects.

Classes and Objects

- Simple example:

```
>>> class MyCourse:
...       name = "NLP"
...       def function(self): #called method
...           print "I love NLP."
```

- Assign MyClass to an object:

```
>>> myObject = MyCourse()
```

Now the variable "myObject" holds an object of the class "MyCourse" that contains the variable and the function defined within the class called "MyCourse".

Classes and Objects

- Accessing elements of an object

```
>>> myObject.name
'NLP'
```

```
>>> myOtherObject = MyCourse()
>>> myOtherObject.name = "comp599"
>>> print myOtherObject.name
comp599
```

Numpy

- Fundamental package for scientific computing with Python.
- Has a powerful N-dimensional array object

```
>>> import numpy as np
>>> x = np.array([[1,2,3],[4,5,6]],np.int32)
array([[1, 2, 3],
       [4, 5, 6]])
```

- Many useful functions

Numpy

- Array Slicing
  - Generate views of the data
  - Format: start : stop : step

```
>>> import numpy as np
>>> x = np.array([[1,2,3],[4,5,6]],np.int32)
array([[1, 2, 3],
       [4, 5, 6]])
>>> y = x[:,1]
>>> y
array([2, 5])
>>> z = np.array([0,1,2,3,4,5,6,7,8,9])
>>> z[1:7:2]
array([1, 3, 5])
```

Scikit-learn

- Machine Learning package in Python.
- Includes many classification, regression and clustering algorithms.
- Also, includes some datasets.

Example: Linear Regression

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

```
import numpy as np
from sklearn import datasets, linear_model
# Load the diabetes dataset
diabetes = datasets.load_diabetes()

# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# The mean square error
print("Residual sum of squares: %.2f"
% np.mean((regr.predict(diabetes_X_test) - diabetes_y_test) ** 2))
```

NLTK

- Natural Language ToolKit
- Contains useful NLP tools such as stemmers, lemmatizers, parsers with a bunch of corpora

NLTK

- Tokenizers
  - Divide string into lists of substrings.
  - For example, tokenizers can be used to find the words and punctuation in a string:

```
>>> from nltk.tokenize import word_tokenize
>>> s = "Good muffins cost $3.88 in New York. Please buy
me two of them."
>>> word_tokenize(s)
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New',
'York', '.', 'Please', 'buy', 'me', 'two', 'of', 'them',
'.']
```

http://www.nltk.org/_modules/nltk/tokenize.html

NLTK

- Stemmers
  - Remove morphological affixes from words, leaving only the word stem.
- Example: Porter stemmer

```
>>> from nltk.stem.porter import *
>>> stemmer = PorterStemmer()
>>> plurals = ['caresses', 'flies', 'dies', 'mules','denied',
'died', 'agreed', 'owned', 'humbled', 'sized', 'meeting', 'stating',
'siezing', 'itemization', 'sensational', 'traditional', 'reference',
'colonizer', 'plotted']
>>> singles = [stemmer.stem(plural) for plural in plurals]
>>> print(' '.join(singles))
caress fli die mule deni die agre own humbl size meet state siez
item sensat tradit refer colon plot
```

http://www.nltk.org/howto/stem.html

NLTK

- Lemmatizers
  - Determine the lemma of words
- Example: WordNet Lemmatizer

```
>>> from nltk.stem import WordNetLemmatizer
>>> wnl = WordNetLemmatizer()
>>> words = ['dogs', 'churches', 'aardwolves', 'abaci',
'hardrock']
>>> lemmata = [wnl.lemmatize(word) for word in words]
>>> for lemma in lemmata: print lemma
dog
church
aardwolf
abacus
hardrock
```

http://www.nltk.org/_modules/nltk/stem/wordnet.html

Final Notes

- Acknowledgment: The first part of the lecture was presented at COLING 2016. Check out the paper on my website!
- Check out the tutorial, "*An intro to Applied Machine Learning in Python*", by fellow RL-labber Pierre-Luc Bacon.