Inglorious LSTMs
Generating Tarantino Movies with Neural Networks

Bohdan Andrusyak
Data Scientist at Know Center
linkedin.com/in/bandrusyak

Today's Questions

- What is NLP?
- Language Models and Who Needs Them?
- What is better Human Brain or Neural Network?
- Are LSTM similar to LSD?
- How can You do All of This in Python?
- How can You Generate Movie Script?

What is NLP?

- NLP - ~~Neuro Linguistic Programing~~ Natural Language Processing
- Sub-field of **Artificial Intelligence** focused on enabling computers to understand and process human languages
- Applications:
  - Language Translation
  - Sentiment Detection
  - Text Categorization
  - Text Summarization
  - Text Generation

Why do computers can not understand humans?

Formal Language:

```
a = 10
b = 20
while a < 20:
    a += 1
```

Human Language:

He is literally on fire

Bring me that thing

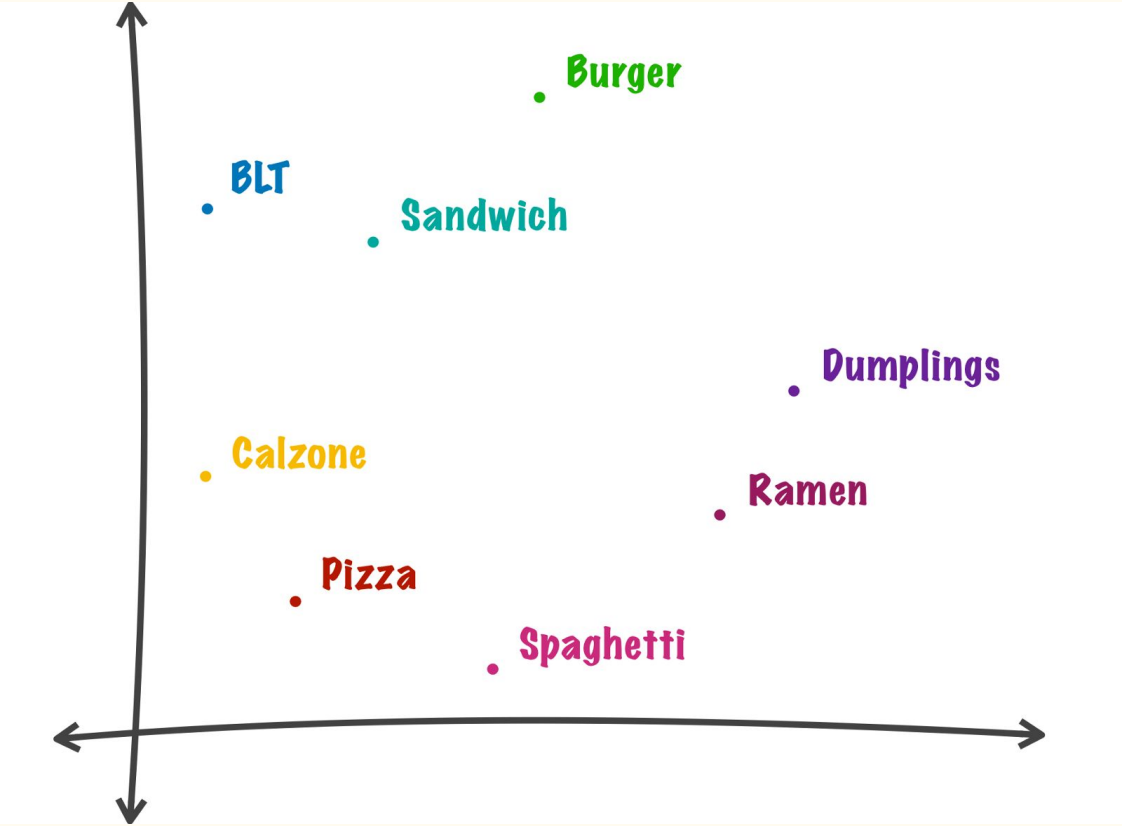
Great job

Language Models what are they?

- Statistical Language Model - probabilistic model that are able to predict the next word in the sequence based on previous words.

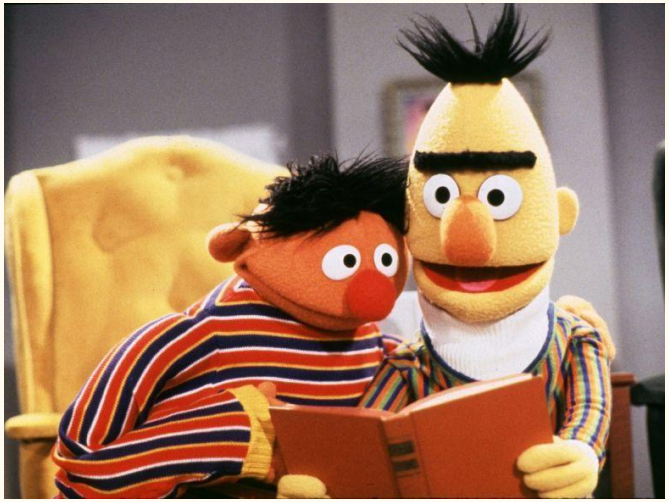
- Neural Language Model - parametrization of words as vectors (word embeddings) and using them as inputs to a neural network. Parameters are learned during training. Words with same meaning are close in vector space.

Word Embeddings Real Life Example

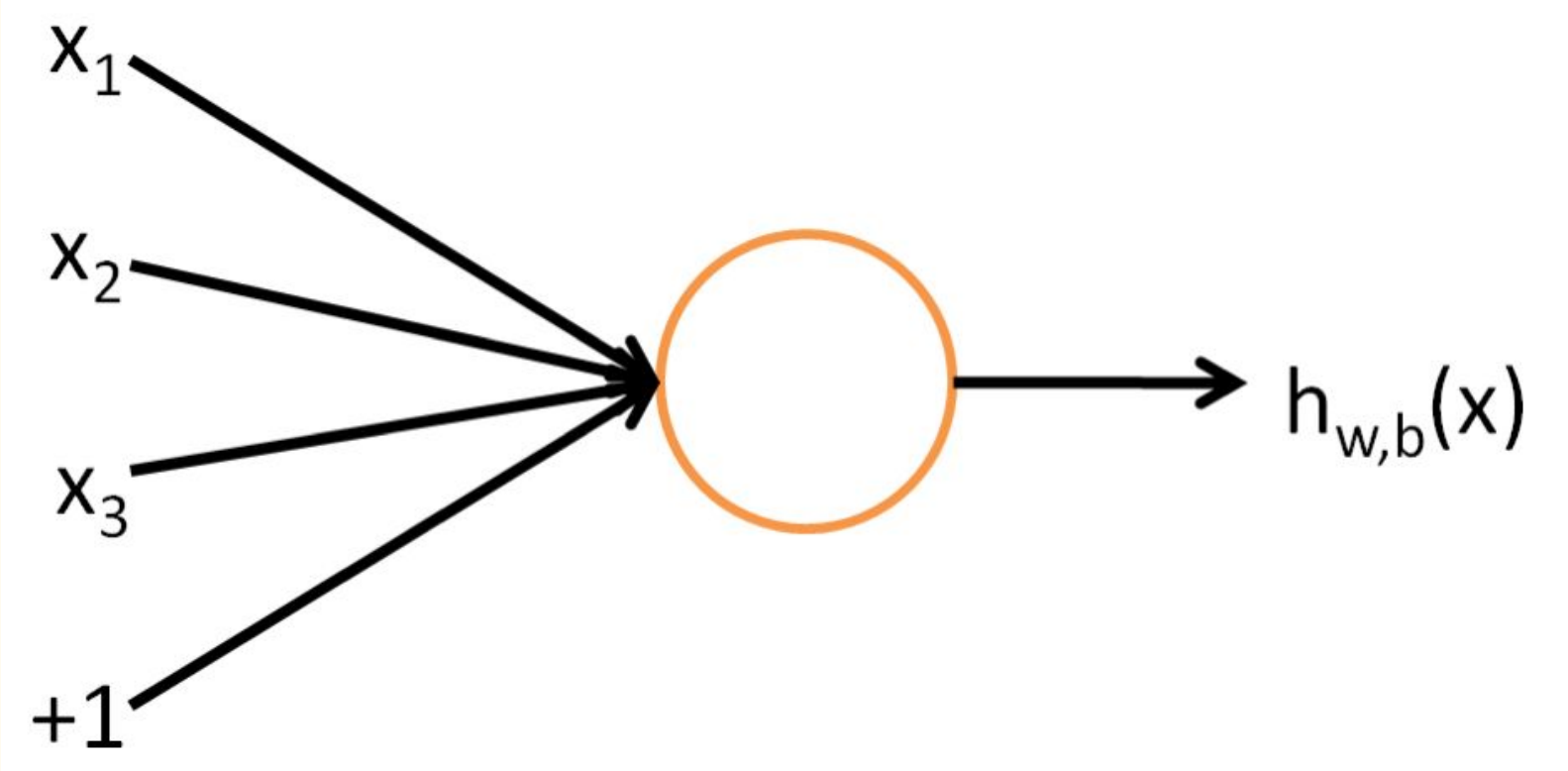


Who uses Language Models?

- Word2vec, Google
- BERT, Google
- ELMo, Allen Institute
- GloVe, Stanford
- fastText, Facebook
- GPT2, Open AI (too dangerous to be released)

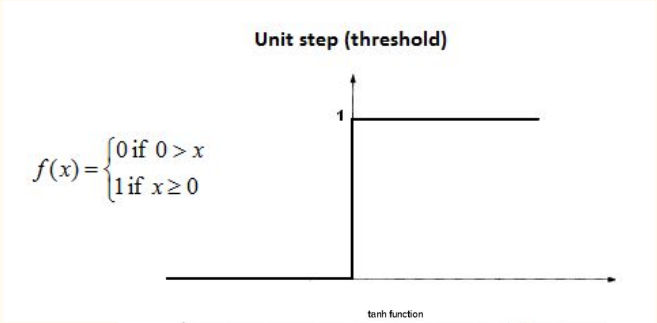


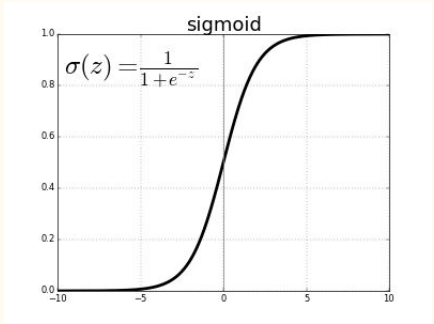
Neuron

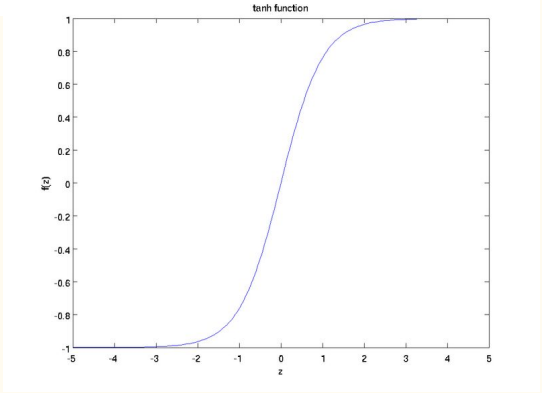


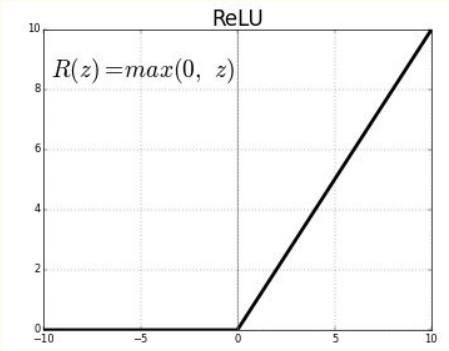
Activation functions

- Step Function
- Sigmoid Function
- Tanh Function
- ReLU Function

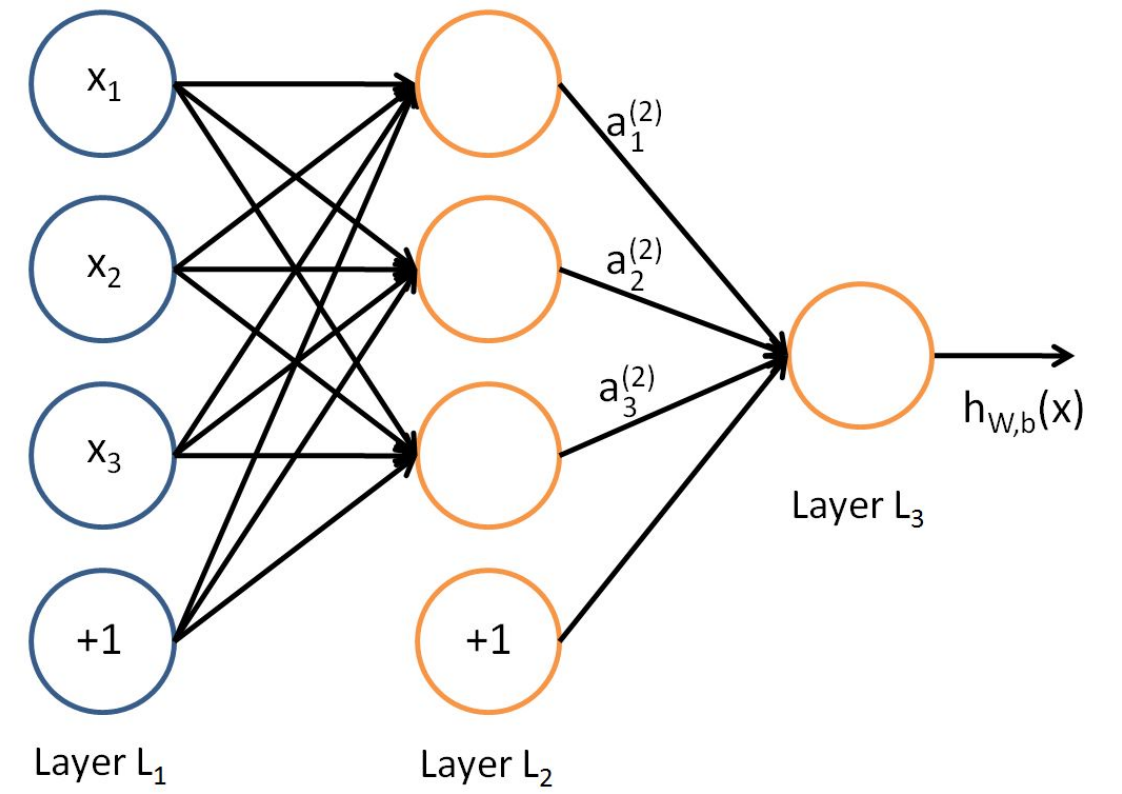




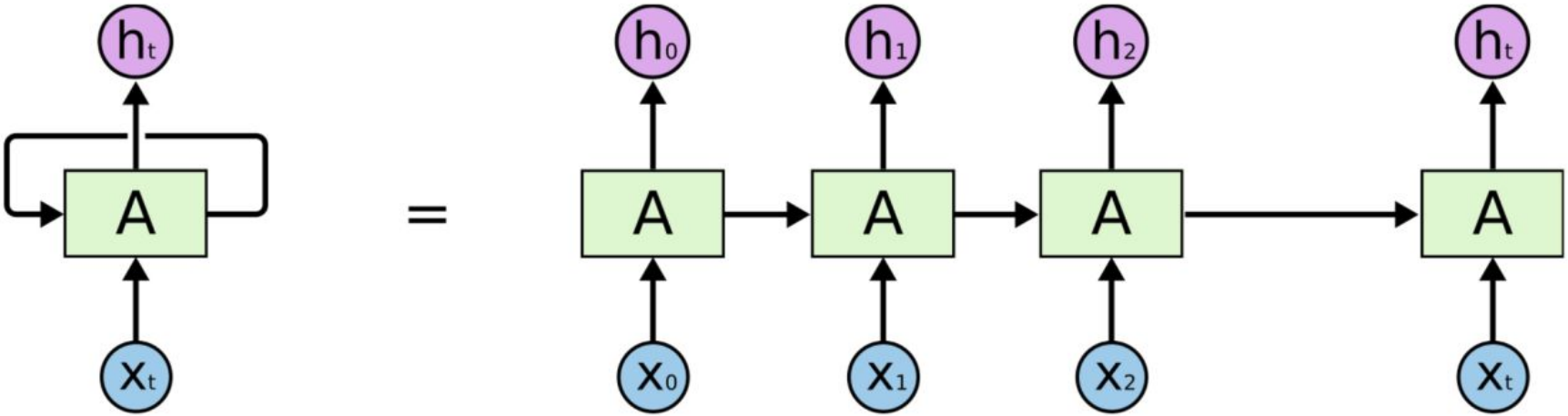




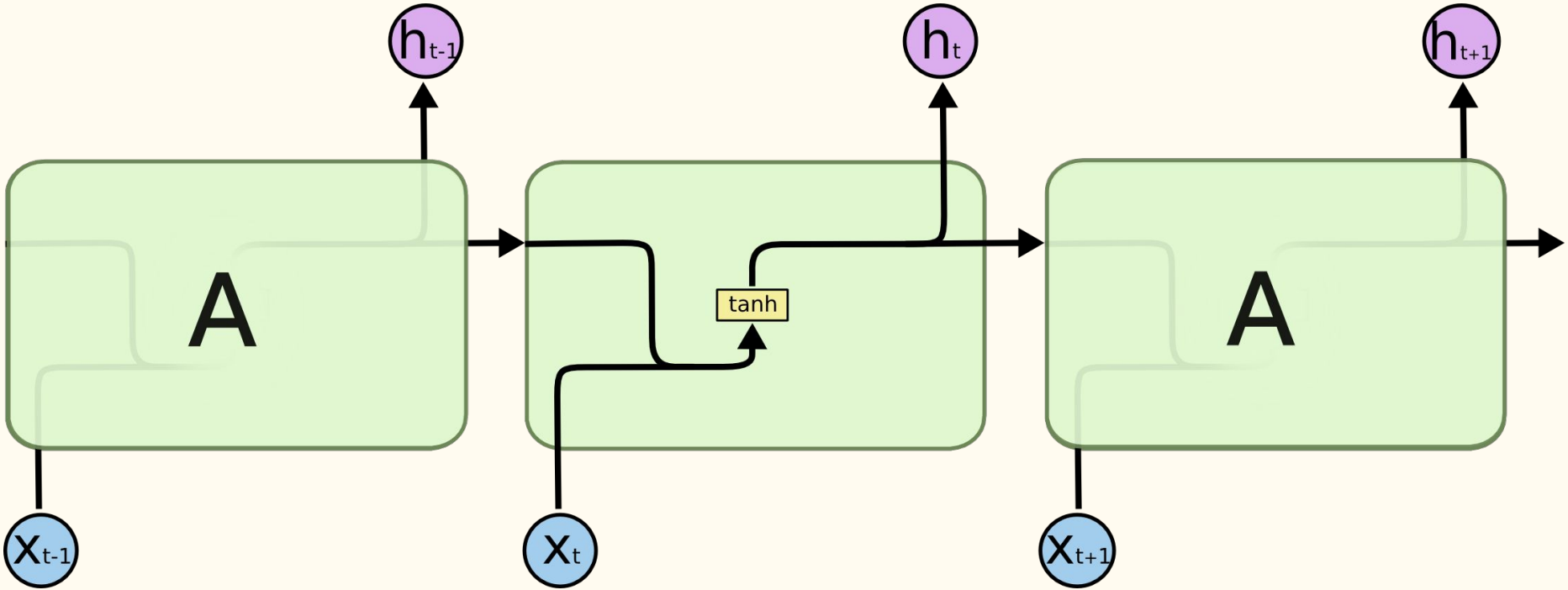
Neural Network



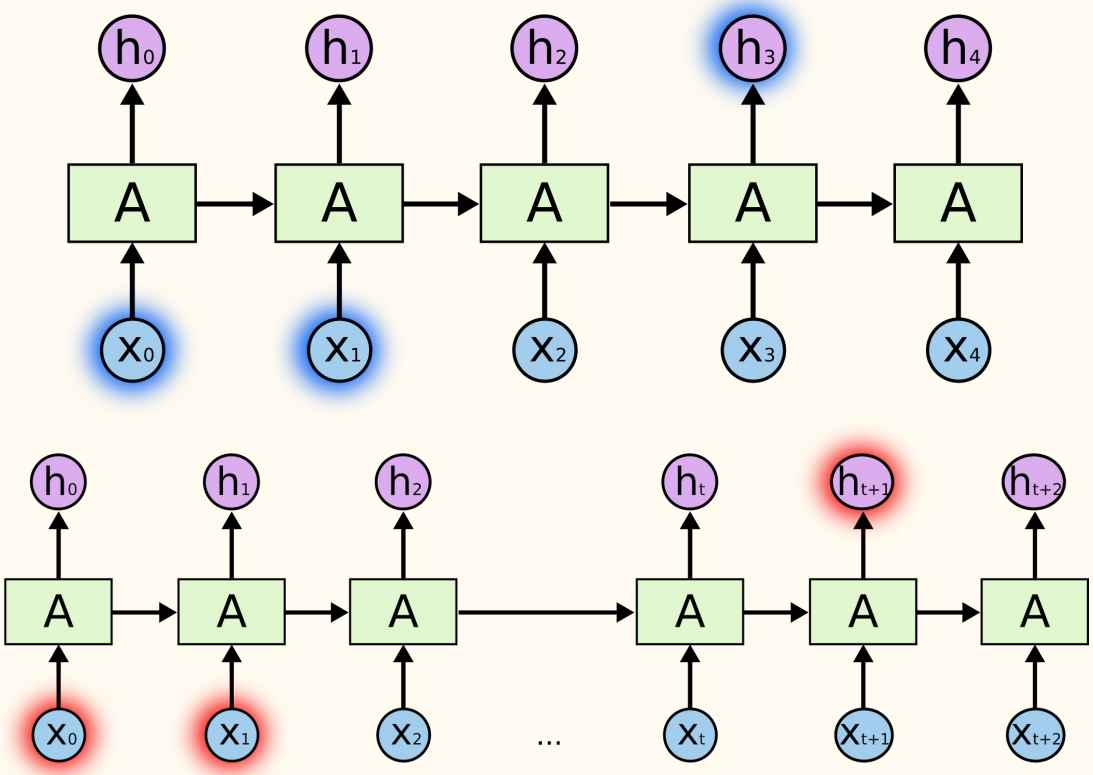
Recurrent Neural Networks



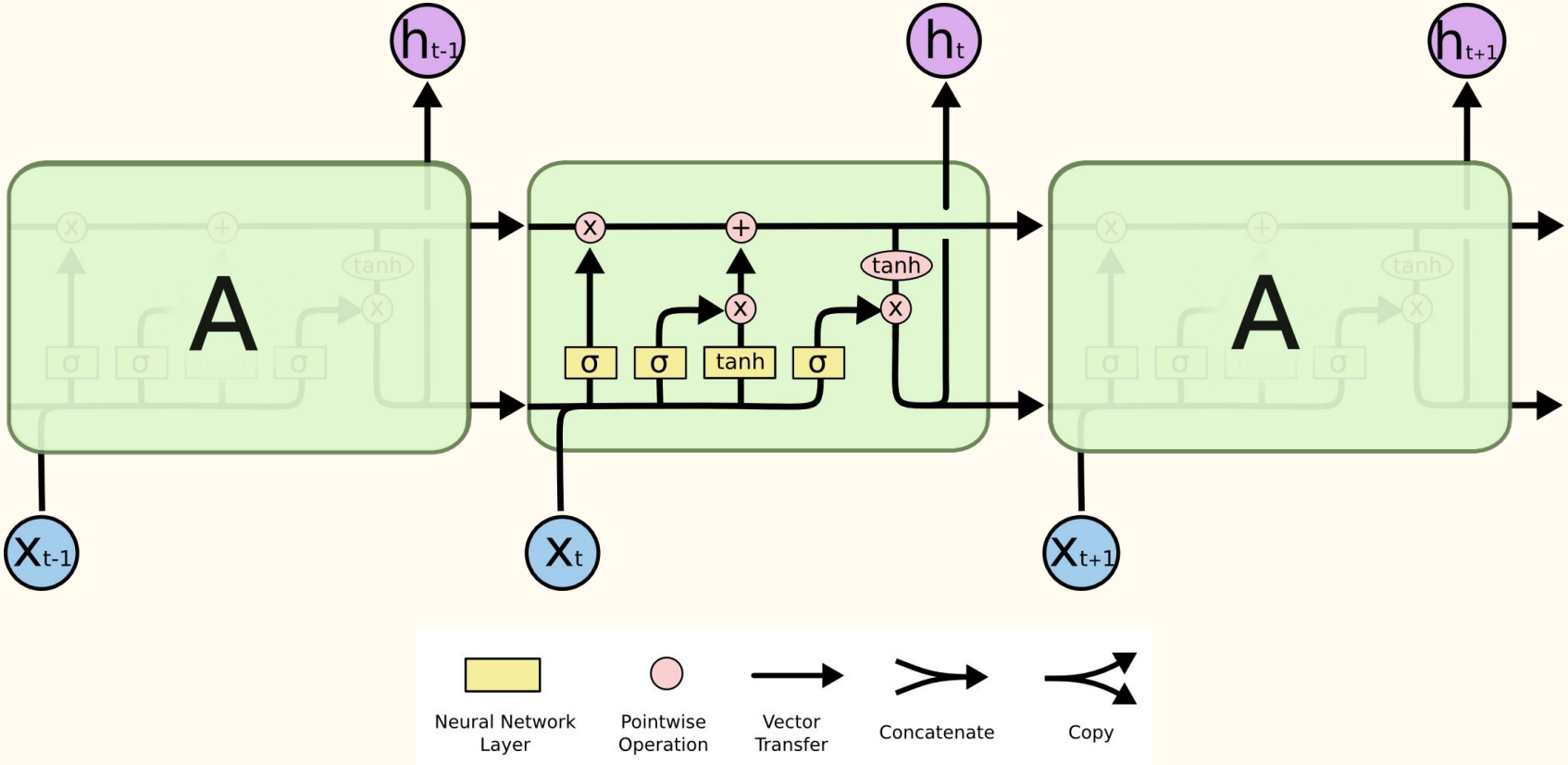
Repeating module in RNN



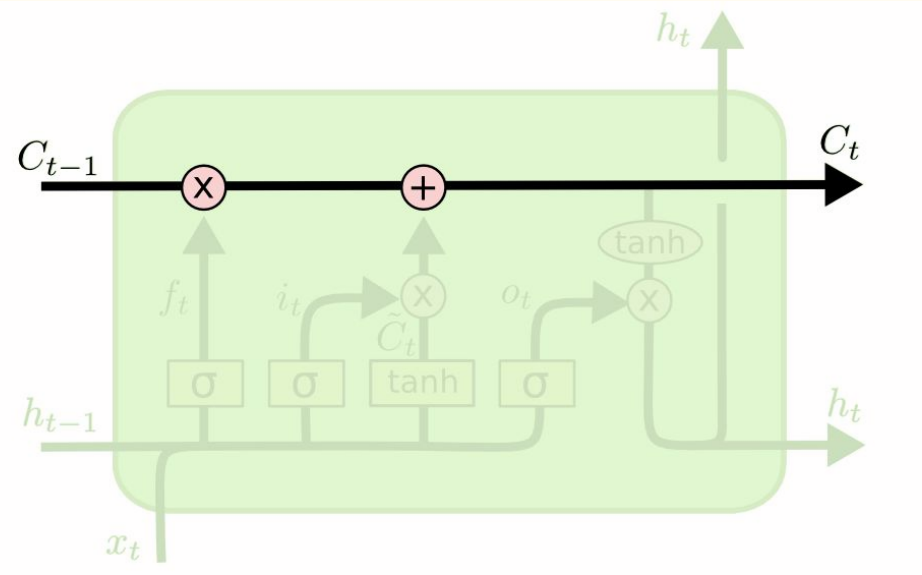
The Problem of Long-Term Dependencies

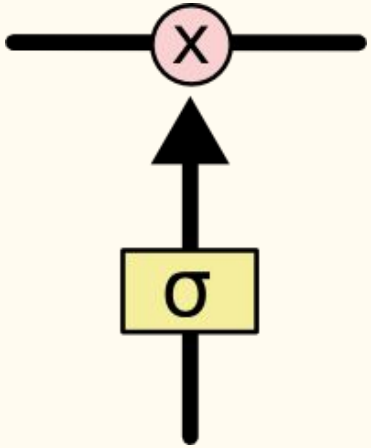


LSTM - Long Short Term Memory

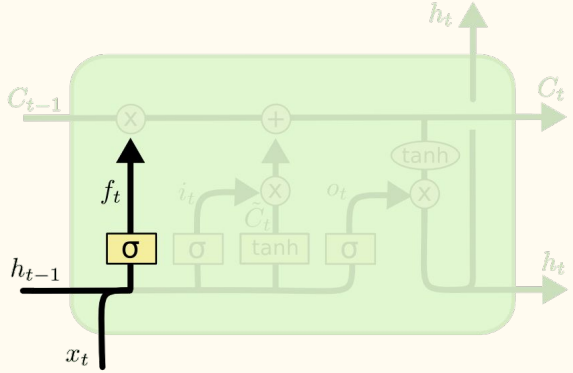


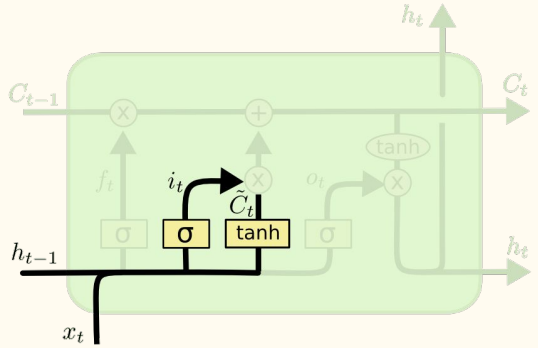
LSTM gates

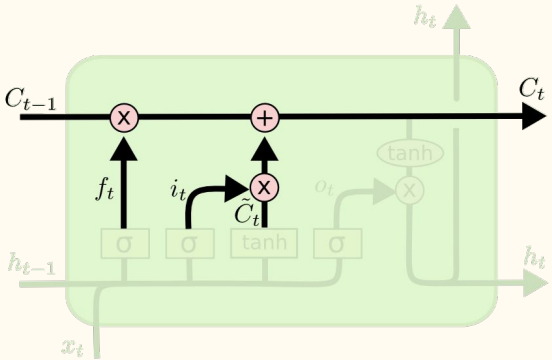


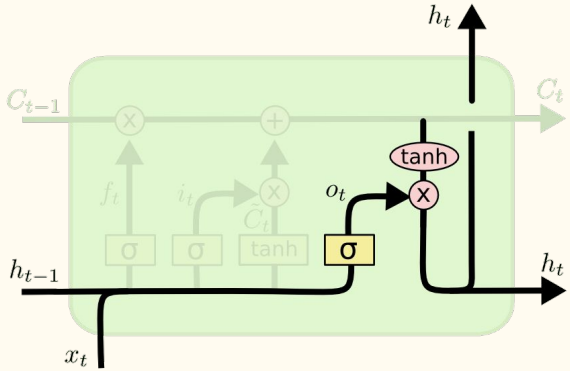


LSTM step by step









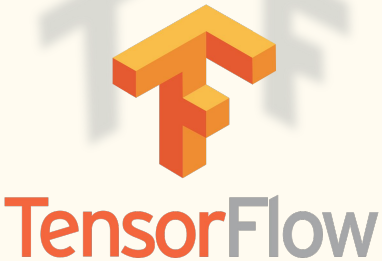
How Can You Do It in Python?

NLP libraries:

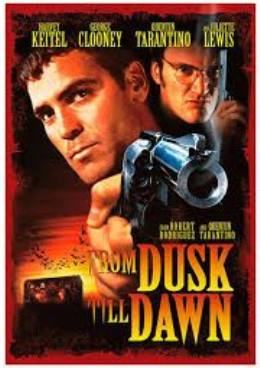
- nltk - natural language toolkit
- spaCy
- gensim

Deep learning:

- TensorFlow
- Keras

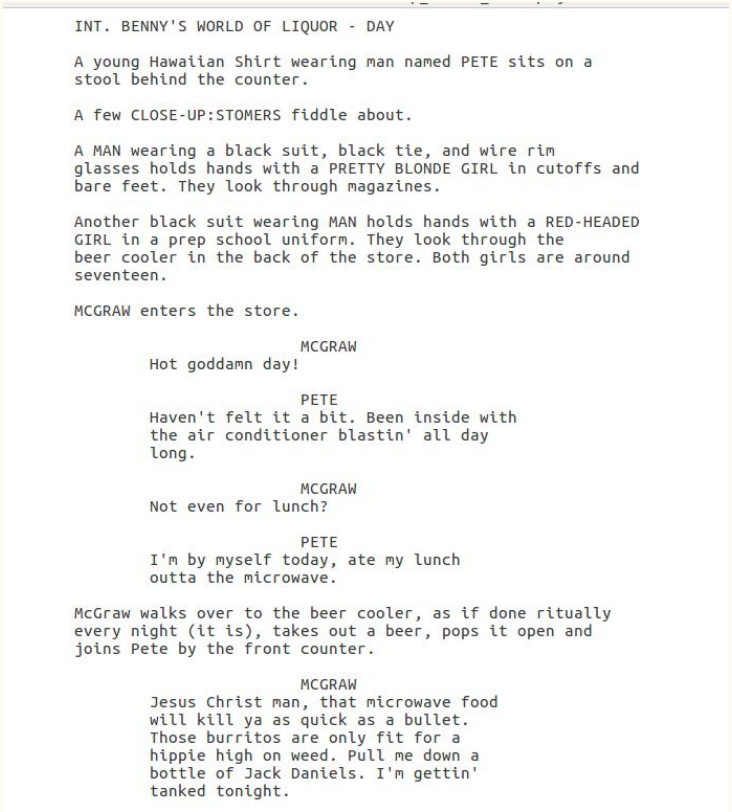


Finally, some practical stuff



Data collection

- Source: imsdb.com
- Movie scripts collected:
  - Natural Born Killers
  - Reservoir Dogs
  - From Dusk till Dawn
  - Pulp Fiction
  - Jackie Brown
  - Kill Bill vol. 1&2
  - Inglorious Bastards
  - Django Unchained



INT. BENNY'S WORLD OF LIQUOR - DAY

A young Hawaiian Shirt wearing man named PETE sits on a stool behind the counter.

A few CLOSE-UP:STOMERS fiddle about.

A MAN wearing a black suit, black tie, and wire rim glasses holds hands with a PRETTY BLONDE GIRL in cutoffs and bare feet. They look through magazines.

Another black suit wearing MAN holds hands with a RED-HEADED GIRL in a prep school uniform. They look through the beer cooler in the back of the store. Both girls are around seventeen.

MCGRAW enters the store.

MCGRAW
Hot goddamn day!

PETE
Haven't felt it a bit. Been inside with the air conditioner blastin' all day long.

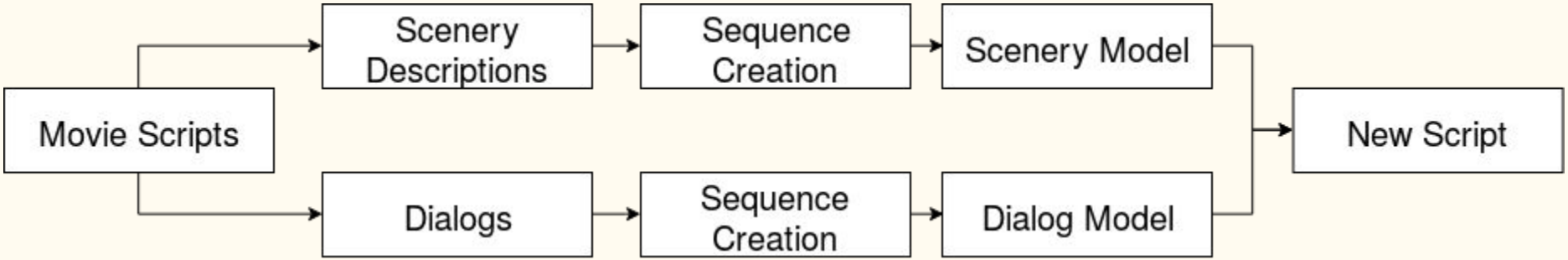
MCGRAW
Not even for lunch?

PETE
I'm by myself today, ate my lunch outta the microwave.

McGraw walks over to the beer cooler, as if done ritually every night (it is), takes out a beer, pops it open and joins Pete by the front counter.

MCGRAW
Jesus Christ man, that microwave food will kill ya as quick as a bullet. Those burritos are only fit for a hippie high on weed. Pull me down a bottle of Jack Daniels. I'm gettin' tanked tonight.

Approach


Movie Scripts
Scenery Descriptions
Sequence Creation
Scenery Model
Dialogs
Sequence Creation
Dialog Model
New Script

Data preparation

```
def extract_dialog_scenery(script):
    script = script.replace(dialog_start, '<CENTER>') #replace whitespaces with placeholder
    script = script.replace('\\n\\n', '<BREAK>') #replace \\n\\n with placeholder
    dialog_patern = '<CENTER>(.*?)<BREAK>'
    raw_dialogs = re.findall('<CENTER>(.*?)<BREAK>', script, re.DOTALL) #find all dialog lines
    script = re.sub('<CENTER>(.*?)<BREAK>', '<DIALOG>', script) #replace all dialog lines with <DIALOG>
    raw_scenary = script.split('<DIALOG>') #divide  scenaries
    clean_dialogs = tokenize_clean(raw_dialogs)
    clean_dialogs = [['<NEW_LINE>'] + line for line in clean_dialogs] #Add indication of new line
    clean_scenery = tokenize_clean(raw_scenary)
    return clean_dialogs, clean_scenery
```

```
def tokenize_clean(lines):
    cleaned_lines = []
    for line in lines:
        # first remove all placeholders we used for dividing script
        line = line.replace('<CENTER>', ' ')
        line = line.replace('<BREAK>', ' ')
        line = line.replace('<DIALOG>', ' ')
        line = gensim.utils.simple_preprocess(line) # cleans and tokenizes text
        if len(line) != 0:
            cleaned_lines.append(line)
    return cleaned_lines
```

Sequences

JULES
Say "What" again! C'mon, say "What" again! I dare ya, I double dare ya motherfucker, say "What" one more goddamn time!

Brett is regressing on the spot.

JULES
Now describe to me what Marsellus Wallace looks like!

Brett does his best.

BRETT
Well he's... he's... black -

JULES
- go on!

BRETT
...and he's... he's... bald -

JULES
- does he look like a bitch?!

BRETT
(without thinking)
What?

<New_Line> jules say what again cmon say what again i dare ya i double dare ya motherfucker say what one more goddamn time <New_Line> jules now describe me what marsellus wallace looks like

Sequence length = 10

1. <New_Line> jules say what again cmon say what again i dare
2. jules say what again cmon say what again i dare ya
3. what again cmon say what again i dare ya i double dare ya motherfucker
4. again i dare ya i double dare ya motherfucker say

Sequences in code

```
def make_sequence(tokens, seuqence_len):
    sequences = list()
    for i in range(seuqence_len, len(tokens)):
        # select sequence of tokens
        seq = tokens[i-seuqence_len:i]
        # convert into a line
        line = ' '.join(seq)
        # store
        sequences.append(line)
    return sequences
```

Create Text Generator

```
def create_text_generator(name,lines, expected_sequence_length, embedding_size=50, lstm_size=100, dence_size=100):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    vocab_size = len(tokenizer.word_index) + 1
    print("Vocablulary size: ", vocab_size)

    sequences = tokenizer.texts_to_sequences(lines)
    sequences = [x for x in sequences if len(x) == expected_sequence_length]
    sequences = np.array(sequences)

    X, y = sequences[:,:-1], sequences[:,-1]
    y = to_categorical(y, num_classes=vocab_size)
    seq_length = X.shape[1]

    model = build_model(vocab_size, seq_length, embedding_size, lstm_size, dence_size)
    # compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # fit model
    mc = keras.callbacks.ModelCheckpoint(path_to_weights + name + '_weights{epoch:08d}.h5',
                                         save_weights_only=True, period=50)
    history = model.fit(X, y, batch_size=128, callbacks=[mc], epochs=100)
    return tokenizer, history, model
```

Build model

```
def build_model(vocab_size, sequence_length, embedding_size, lstm_size, dence_size):
    model = Sequential()
    model.add(Embedding(vocab_size, embedding_size, input_length=sequence_length))
    model.add(LSTM(lstm_size, return_sequences=True))
    model.add(LSTM(lstm_size))
    model.add(Dense(dence_size, activation='relu'))
    model.add(Dense(vocab_size, activation='softmax'))
    return model
```

Creating New Texts

```
def generate_seq(model, tokenizer, seq_length, seed_text, n_words):
    result = list()
    in_text = seed_text
    # generate a fixed number of words
    for _ in range(n_words):
        # encode the text as integer
        encoded = tokenizer.texts_to_sequences([in_text])[0]
        # truncate sequences to a fixed length
        encoded = pad_sequences([encoded], maxlen=seq_length, truncating='pre')
        # predict probabilities for each word
        yhat = model.predict_classes(encoded, verbose=0)
        # map predicted word index to word
        out_word = ''
        for word, index in tokenizer.word_index.items():
            if index == yhat:
                out_word = word
                break
        # append to input
        in_text += ' ' + out_word
        result.append(out_word)
    return ' '.join(result)
```

Results

SEED: the path of the righteous man is beset on all sides by the inequities of the selfish and the tyranny of evil men

RESULT: blessed is he who in the name of charity and good will shepherds the weak through the valley of the darkness

More Results

SEED: i ll take care of this you guys leave the tip and when i come back i want my book back

RESULT: where apologize counting the salesgirl roger to drink mr white shit

Even More Results

SEED: a normal dennys spires like coffee shop in los angeles it is about in the morning while the place is not jammed there is a healthy number of people drinking coffee munching on bacon and eating eggs

RESULT: two dogs ext bennett manor day cut the way with white wagon reading from his noodle

Resources

- Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning, https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90
- Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Source code, https://github.com/bohdan1/TarantinoLSTM