Women Machine Learning Bootcamp: Day 5

Deep Learning for Natural Language Processing - RNNs

Instructor: Dr. Rabeah Al-Zaidy

Accreditation:

- Credits for figures and content used in these slides:
  - https://web.stanford.edu/class/archive/cs/cs224n/
  - https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
  - https://machinelearningmastery.com/develop-word-embeddings-python-gensim/
  - https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

We will learn about..

- What is Natural Language Processing ?
- What is Deep Learning (DL)?
- DL for NLP
- Dense Word Vectors (gensim)
- Sentiment Analysis (pytorch)

What is Natural Language Processing (NLP)?

- Natural language processing is a field at the intersection of
  - computer science
  - artificial intelligence
  - and linguistics. •
- Goal: for computers to process or "understand" natural language in order to perform tasks that are useful:
  - Language translation
  - Question Answering
  - Siri, Google Assistant, Facebook M, Cortana …
- Fully understanding and representing the meaning of language (or even defining it) is a difficult goal.
- Perfect language understanding is AI-complete

Why is NLP hard?

- Complexity in representing, learning and using linguistic/ situational/contextual/world/visual knowledge
- But interpretation depends on these
- Human languages are ambiguous (unlike programming and other formal languages)
- E.g. "I made her duck."

A sample of NLP Applications

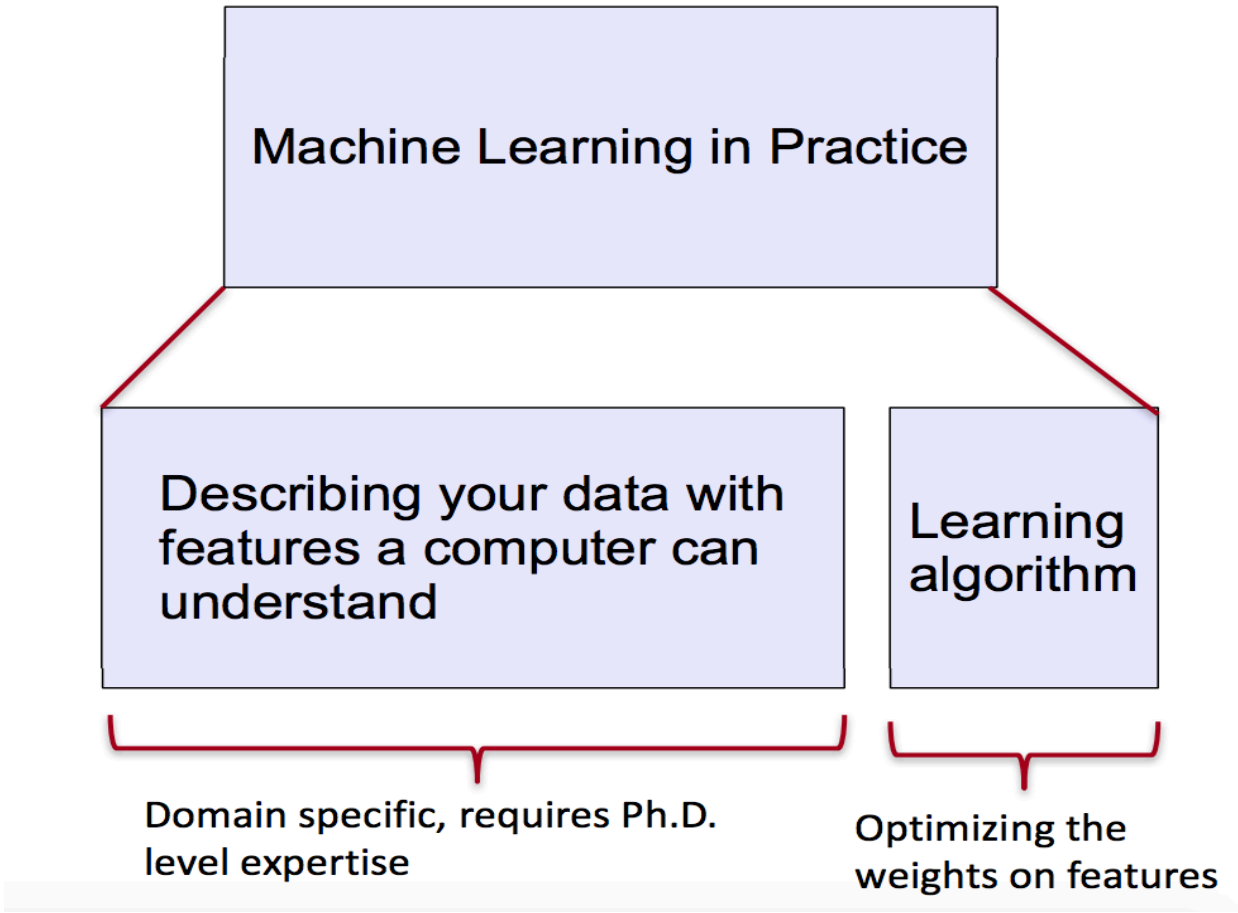
Applications range from simple to complex:

- Spell checking, keyword search, finding synonyms
- Extracting information from websites such as
  - product price, dates, location, people or company names
- Classifying: reading level of school texts, positive/negative sentiment of longer documents
- Machine translation
- Spoken dialog systems
- Complex question answering

NLP in industry

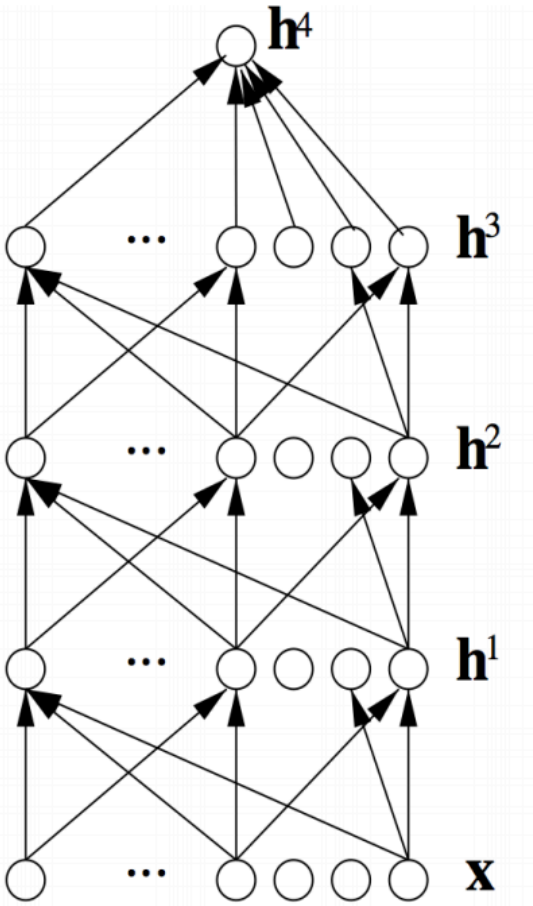
- Search (written and spoken)
- Online advertisement matching
- Automated/assisted translation
- Sentiment analysis for marketing or finance/trading
- Speech recognition
- Chatbots / Dialog agents
  - Automating customer support
  - Controlling devices
  - Ordering goods

Machine Learning vs. Deep Learning



What is Deep Learning (DL)?

- In contrast to standard machine learning
- Representation learning attempts to automatically learn good features or representations
- Deep learning algorithms attempt to learn (multiple levels of) representations (here: h1,h2,h3) and an output (h4)
- From "raw" inputs x (e.g. sound, pixels, characters, or words)



Reasons for Exploring Deep Learning

- Manually designed features are often over-specified, incomplete and take a long time to design and validate
- Learned Features are easy to adapt, fast to learn
- Deep learning provides a very flexible, (almost?) universal, learnable framework for representing world, visual and linguistic information.
- Deep learning can learn unsupervised (from raw text) and supervised (with specific labels like positive/negative)

Reasons for Exploring Deep Learning

- Large amounts of training data favor deep learning
- Faster machines and multicore CPU/GPUs favor Deep Learning
- New models, algorithms, ideas
- Better, more flexible learning of intermediate representations
- Effective end-to-end joint system learning
- Effective learning methods for using contexts and transferring between tasks
- Better regularization and optimization methods and improved performance (first in speech and vision, then NLP)

Deep NLP = Deep Learning + NLP

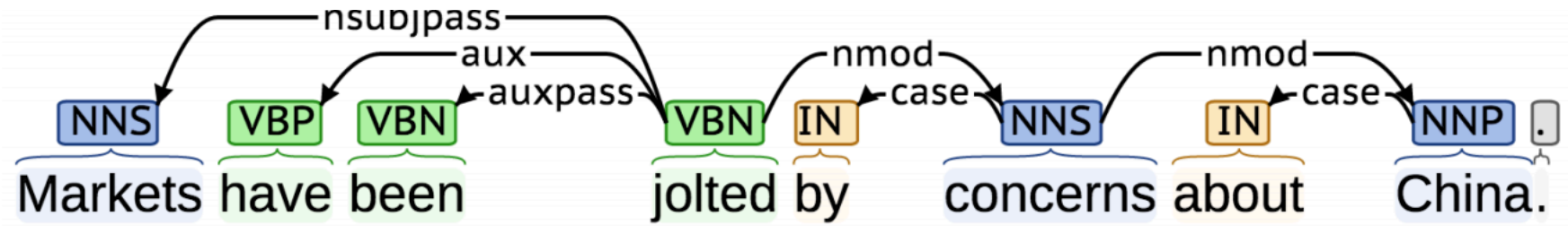
Combine ideas and goals of NLP with using representation learning and deep learning methods to solve them

Several big improvements in recent years in NLP

- Linguistic levels: (speech), words, syntax, semantics
- Intermediate tasks/tools: parts-of-speech, entities, parsing
- Full applications: sentiment analysis, question answering, dialogue agents, machine translation

NLP Tools: Parsing for sentence structure

- Neural networks can accurately determine the grammatical structure of sentences
- This supports interpretation and may help in disambiguation



NLP Applications: Sentiment Analysis

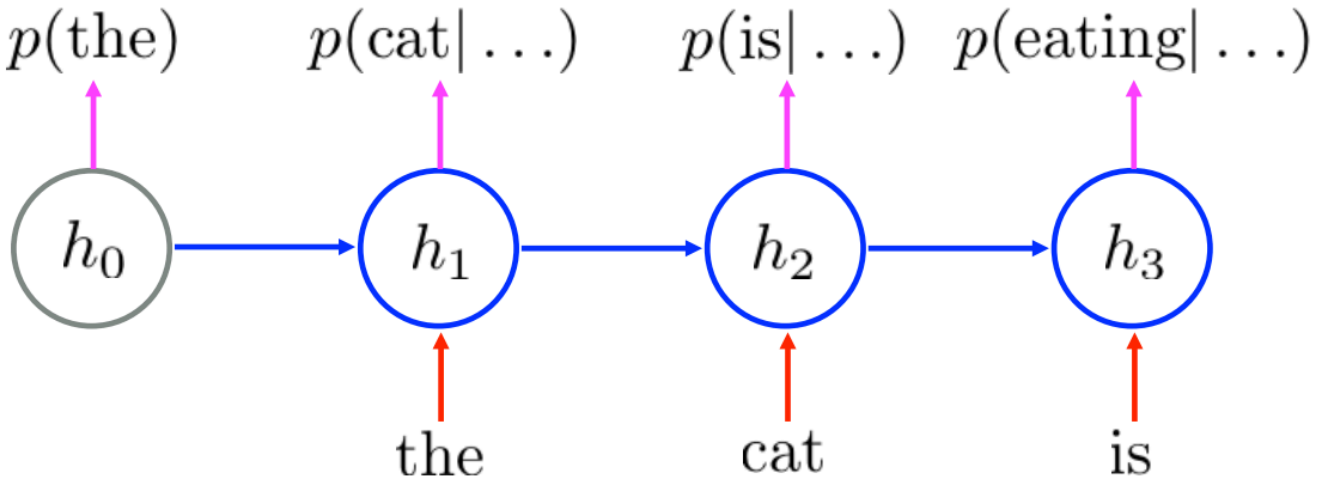
- Traditional: Treat sentence as a bag-of-words (ignore word order); consult a curated list of "positive" and "negative" words to determine sentiment of sentence. Need hand-designed features to capture negation
- Deep learning: using variants of RNNS

Question Answering

- Traditional: A lot of feature engineering to capture world and other knowledge, e.g., regular expressions, Berant et al. (2014)
- DL: Again, a deep learning architecture can be used!
- Facts are stored in vectors

Dialogue agents / Response Generation

- A simple, successful example is the auto-replies available in the Google Inbox app
- An application of the powerful, general technique of Neural Language Models, which are an instance of Recurrent Neural Networks



Word Representation

- Word2vec introduction
- Skip-gram vs CBOW
- Training vs pretrained

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols: hotel, conference, motel

Words can be represented by one-hot vectors:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g. 500,000)

Representing words by their context

- Core idea:
  A word's meaning is given by the words that frequently appear close-by

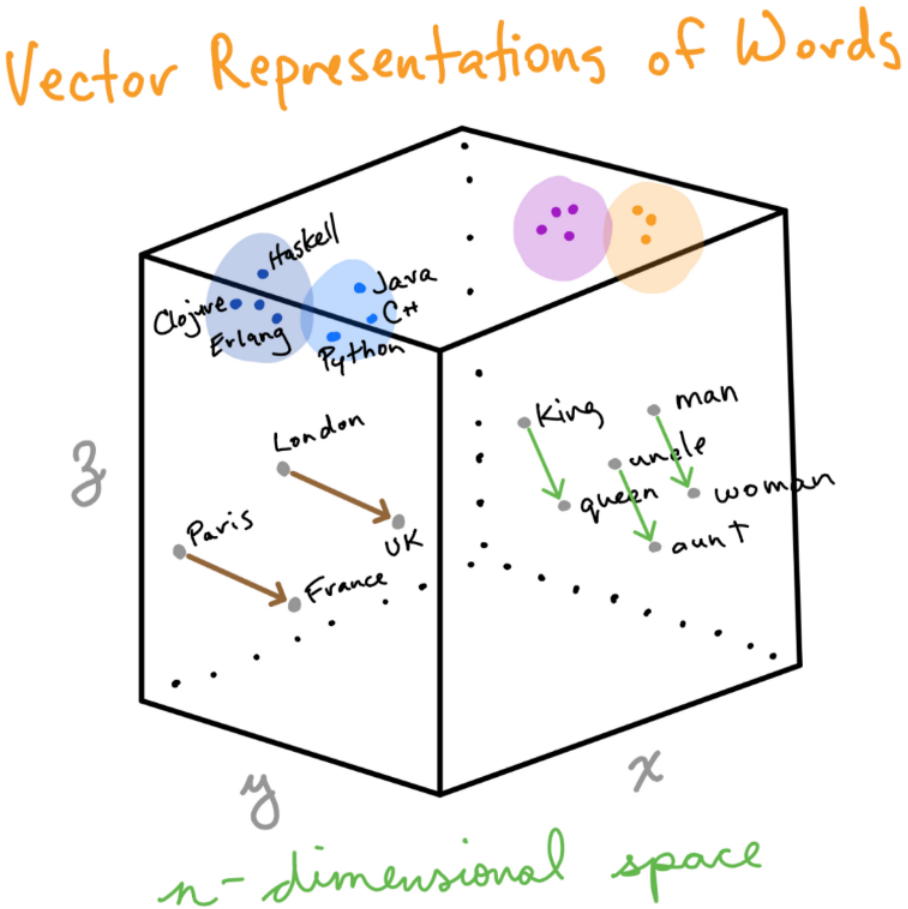
You shall know a word by the company it keeps" (J. R. Firth 1957: 11)

- One of the most successful ideas of modern statistical NLP!
- When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of w to build up a representation of w

… debt problems turning into **banking** crises as happened in 2009…
…saying that Europe needs a unified **banking** regulation to replace the hodgepodge…
…India has just given its **banking** system a shot in the arm…

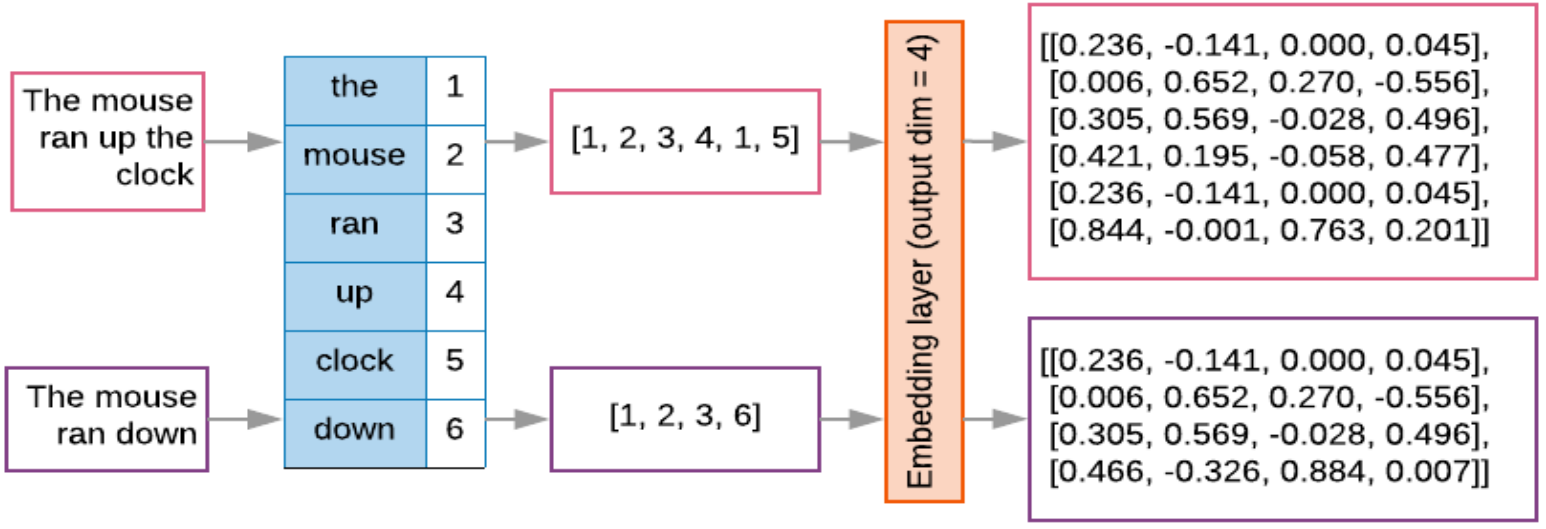
Context

Word meaning as a dense real-valued vector



Word vectors

- A dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.
- Word vectors are sometimes called word embeddings or word representations.

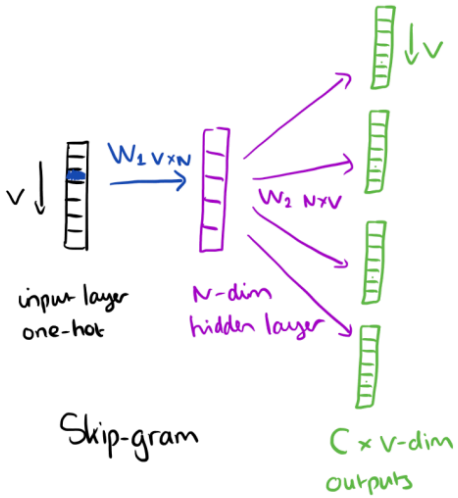


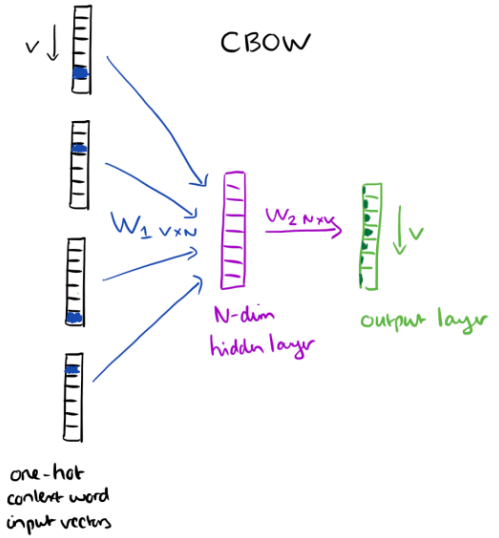
Word2vec: Overview

- Word2vec (Mikolov et al. 2013) is a framework for learning word vectors.
- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word **c** and context ("outside") words **o**
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Word2vec: Overview

- Skip-grams (SG) Predict context ("outside") words (position independent) given center word
- Continuous Bag of Words (CBOW) Predict center word from (bag of) context words





Developing Word Embeddings in Python with Gensim

1. Generating Word Embeddings
    1. Gensim Library
    2. Develop Word2Vec Embedding
2. Visualize Word Embedding
3. Load Stanford's GloVe Embedding

1.Genering Word2Vec Embeddings

- Text must be tokenized:
  - "This is the first sentence." -> ["This", "is", "the", "first", "sentence"]
- Create a Word2Vec instance. Specify:
  - **size**: (default 100) The number of dimensions of the embedding, e.g. the length of the dense vector to represent each token (word).
  - **window**: (default 5) The maximum distance between a target word and words around the target word.
  - **min_count**: (default 5) The minimum count of words to consider when training the model; words with an occurrence less than this count will be ignored.
  - **workers**: (default 3) The number of threads to use while training.
  - **sg**: (default 0 or CBOW) The training algorithm, either CBOW (0) or skip gram (1).
- Save the model (our embeddings!) to disk. Specify: file name, binary or ASCII
- Load our model.

2. Visualizing Word Embeddings

- Retrieve all of the embedding vectors from a **trained** model
- Project n-dimensional vectors to 2-D:
  - PCA model: Principle Component Analysis projector
- Plot the 2-D vector as (x,y) points in the plane
  - Scatter plot is a good way to see the points

Loading Stanford's GloVe Embedding

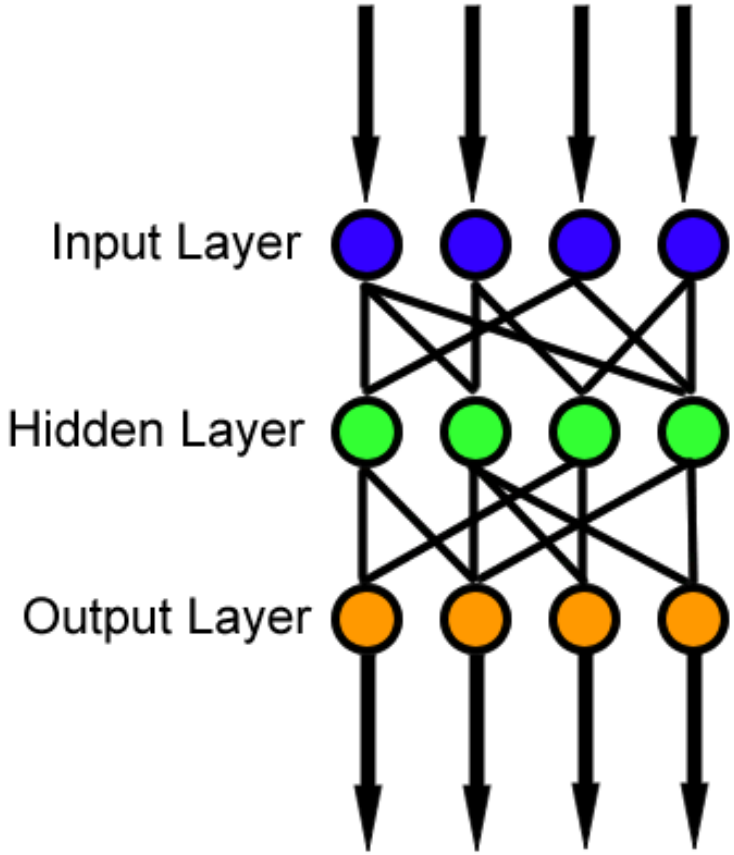
- Wikipedia 2014 + Gigaword 5:
  6B tokens,
  - 400K vocab,
  - uncased,
  - 50d, 100d, 200d,
  - 300d vectors, 822 MB glove.6B.zip
- Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip

Recurrent Neural Networks

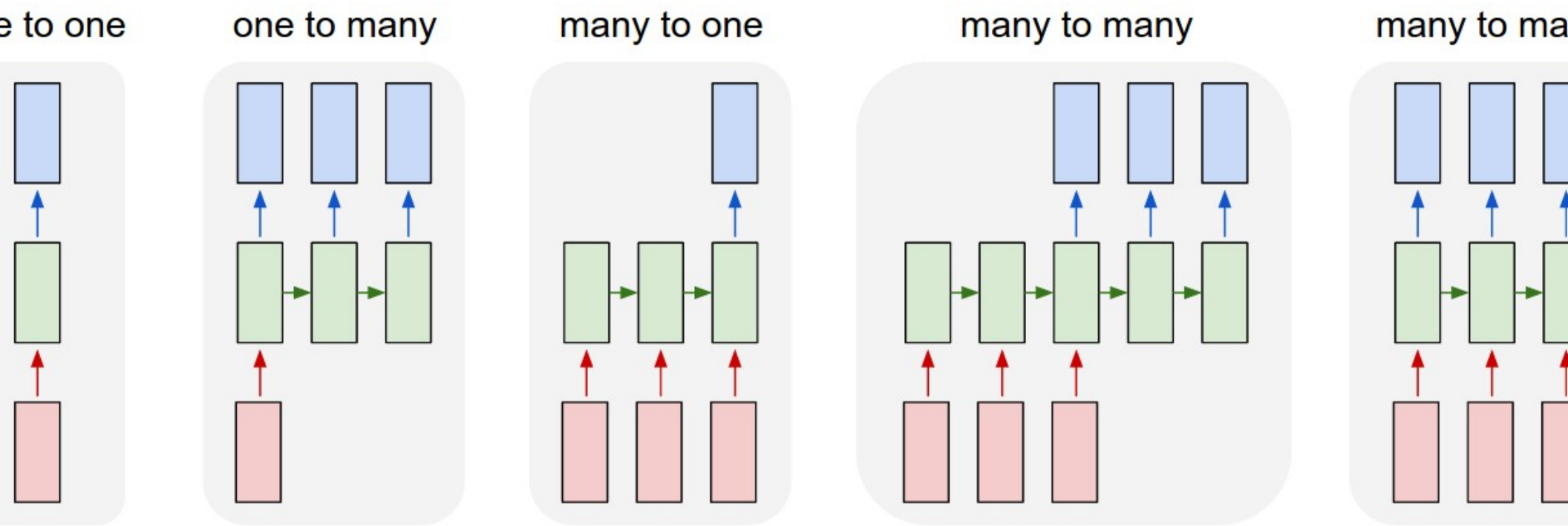
- Feedforward Networks vs Recurrent Networks
- RNNs for Sentiment Analysis
- How to implement an RNN in pytorch
- Training an RNN for SA on the IMDb movie dataset

Feedforward networks

- Information moves in only one direction – forward – from the input nodes, through the hidden nodes and to the output nodes.
- There are no cycles or loops in the network.
- Examples: Multi layer Perceptron

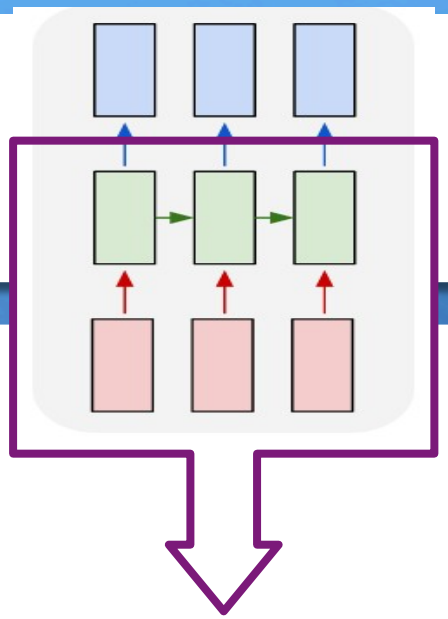


Recurrent Neural Networks



RNN Structure

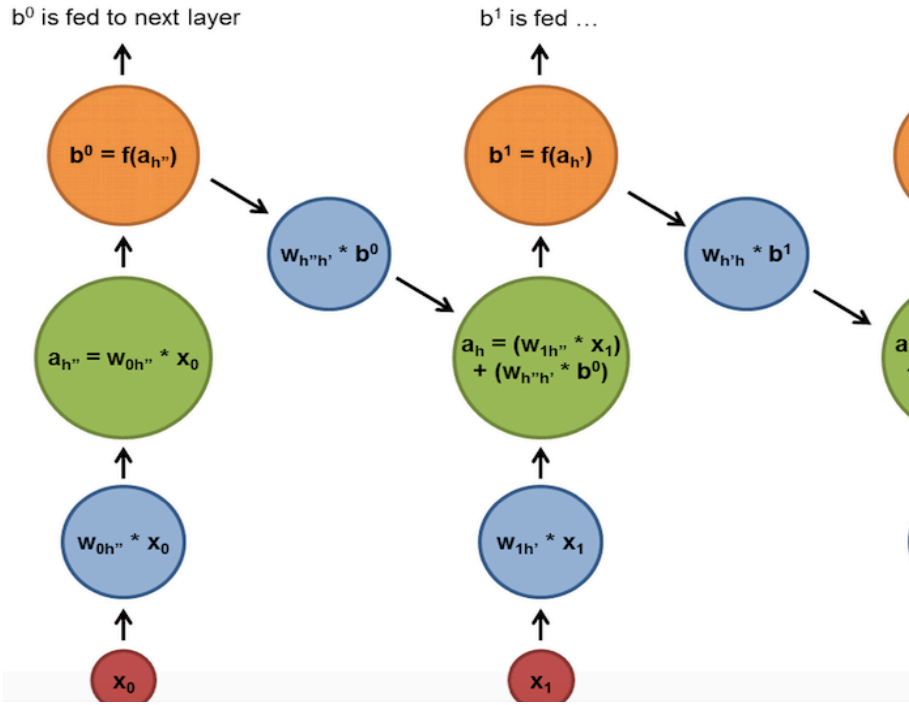
- Takes input vector x, gives output vector y



- Computes hidden layer for time-step t using x_t and h_{t-1}

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

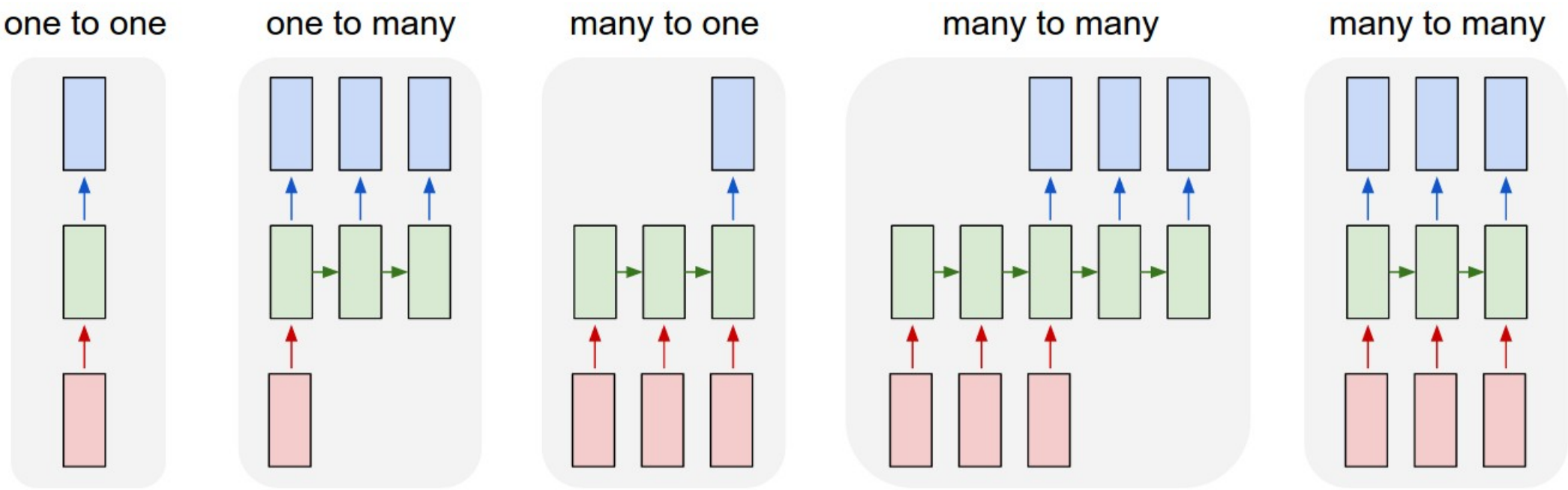
- Computes y using output function



What is Sentiment Analysis?

- Sentiment Analysis/Opinion Mining is a field within Natural Language Processing (NLP) that builds systems that try to identify and extract opinions within text.
- Technically, we extract attributes of the expression:
  - **Polarity**: if the speaker expresses a positive or negative opinion
  - **Subject**: the thing that is being talked about
  - **Opinion holder**: the person, or entity that expresses the opinion

RNN for Sentiment Analysis



Implementing Binary SA Classifier in Pytorch

- Data preprocessing
- Model design
- Training the model
- Testing the model