Natural Language Processing

Transforms human language into machine-usable code

Processing Techniques

- Tokenization - splits text into individual words (tokens)
- Lemmatization - reduces words to its base form based on dictionary definition (*am, are, is* $\rightarrow$ *be*)
- Stemming - reduces words to its base form without context (*ended* $\rightarrow$ *end*)
- Stop words - removes common and irrelevant words (*the, is*)

Markov Chain - stochastic and memoryless process that predicts future events based only on the current state

***n*-gram** - predicts the next term in a sequence of n terms based on Markov chains

Bag-of-words - represents text using word frequencies, without context or order

tf-idf - measures word importance for a document in a collection (corpus), by multiplying the term frequency (occurrences of a term in a document) with the inverse document frequency (penalizes common terms across a corpus)

Cosine Similarity - measures similarity between vectors, calculated as $\cos(\theta) = \frac{A \cdot B}{||A|| ||B||}$, which ranges from 0 to 1

Word Embedding

Maps words and phrases to numerical vectors

word2vec - trains iteratively over local word context windows, places similar words close together, and embeds sub-relationships directly into vectors, such that $king - man + woman \approx queen$

Relies on one of the following:

- Continuous bag-of-words (CBOW) - predicts the word given its context
- skip-gram - predicts the context given a word

GloVe - combines both global and local word co-occurence data to learn word similarity

BERT - accounts for word order and trains on subwords, and unlike word2vec and GloVe, BERT outputs different vectors for different uses of words (*cell* phone vs. blood *cell*)

Sentiment Analysis

Extracts the attitudes and emotions from text

Polarity - measures positive, negative, or neutral opinions

- Valence shifters - capture amplifiers or negators such as '*really* fun' or '*hardly* fun'

Sentiment - measures emotional states such as happy or sad

Subject-Object Identification - classifies sentences as either subjective or objective

Topic Modelling

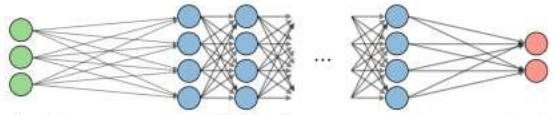
Captures the underlying themes that appear in documents

Latent Dirichlet Allocation (LDA) - generates k topics by first assigning each word to a random topic, then iteratively updating assignments based on parameters α, the mix of topics per document, and β, the distribution of words per topic

Latent Semantic Analysis (LSA) - identifies patterns using tf-idf scores and reduces data to k dimensions through SVD

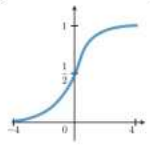
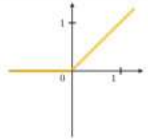
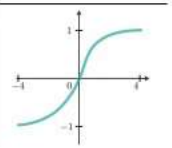
Neural Network

Feeds inputs through different hidden layers and relies on weights and nonlinear functions to reach an output



Perceptron - the foundation of a neural network that multiplies inputs by weights, adds bias, and feeds the result z to an activation function

Activation Function - defines a node's output

Sigmoid	ReLU	Tanh
$\frac{1}{1+e^{-z}}$	$\max(0, z)$	$\frac{e^z - e^{-z}}{e^z + e^{-z}}$
		

Softmax - given final layer outputs, provides class probabilities that sum to 1 $\rightarrow \frac{e^{z_i}}{\sum e^z}$

If there is more than one 'correct' label, the sigmoid function provides probabilities for all, some, or none of the labels.

Loss Function - measures prediction error using functions such as MSE for regression and binary cross-entropy for probability-based classification

Gradient Descent - minimizes the average loss by moving iteratively in the direction of steepest descent, controlled by the learning rate γ (step size). Note, γ can be updated adaptively for better performance. For neural networks, finding the best set of weights involves:

1. Initialize weights W randomly with near-zero values
2. Loop until convergence:
   - Calculate the average network loss $J(W)$
   - **Backpropagation** - iterate backwards from the last layer, computing the gradient $\frac{\partial J(W)}{\partial W}$ and updating the weight $W \leftarrow W - \gamma \frac{\partial J(W)}{\partial W}$
3. Return the minimum loss weight matrix W

To prevent overfitting, regularization can be applied by:

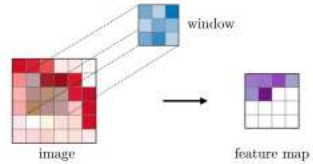
- Stopping training when validation performance drops
- Dropout - randomly drop some nodes during training to prevent over-reliance on a single node
- Embedding weight penalties into the objective function
- Batch Normalization - stabilizes learning by normalizing inputs to a layer

Stochastic Gradient Descent - only uses a single point to compute gradients, leading to smoother convergence and faster compute speeds. Alternatively, mini-batch gradient descent trains on small subsets of the data, striking a balance between the approaches.

Convolutional Neural Network

Analyzes structural or visual data by extracting local features

Convolutional Layers - iterate over windows of the image, applying weights, bias, and an activation function to create feature maps. Different weights lead to different features maps.



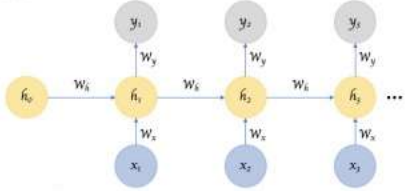
Pooling - downsamples convolution layers to reduce dimensionality and maintain spatial invariance, allowing detection of features even if they have shifted slightly. Common techniques return the max or average value in the pooling window.

The general CNN architecture is as follows:

1. Perform a series of convolution, ReLU, and pooling operations, extracting important features from the data
2. Feed output into a fully-connected layer for classification, object detection, or other structural analyses

Recurrent Neural Network

Predicts sequential data using a temporally connected system that captures both new inputs and previous outputs using hidden states



RNNs can model various input-output scenarios, such as many-to-one, one-to-many, and many-to-many. Relies on parameter (weight) sharing for efficiency. To avoid redundant calculations during backpropagation, downstream gradients are found by chaining previous gradients. However, repeatedly multiplying values greater than or less than 1 leads to:

- Exploding gradients - model instability and overflows
- Vanishing gradients - loss of learning ability

This can be solved using:

- Gradient clipping - cap the maximum value of gradients
- ReLU - its derivative prevents gradient shrinkage for $x > 0$
- Gated cells - regulate the flow of information

Long Short-Term Memory - learns long-term dependencies using gated cells and maintains a separate cell state from what is outputted. Gates in LSTM perform the following:

1. Forget and filter out irrelevant info from previous layers
2. Store relevant info from current input
3. Update the current cell state
4. Output the hidden state, a filtered version of the cell state

LSTMs can be stacked to improve performance.