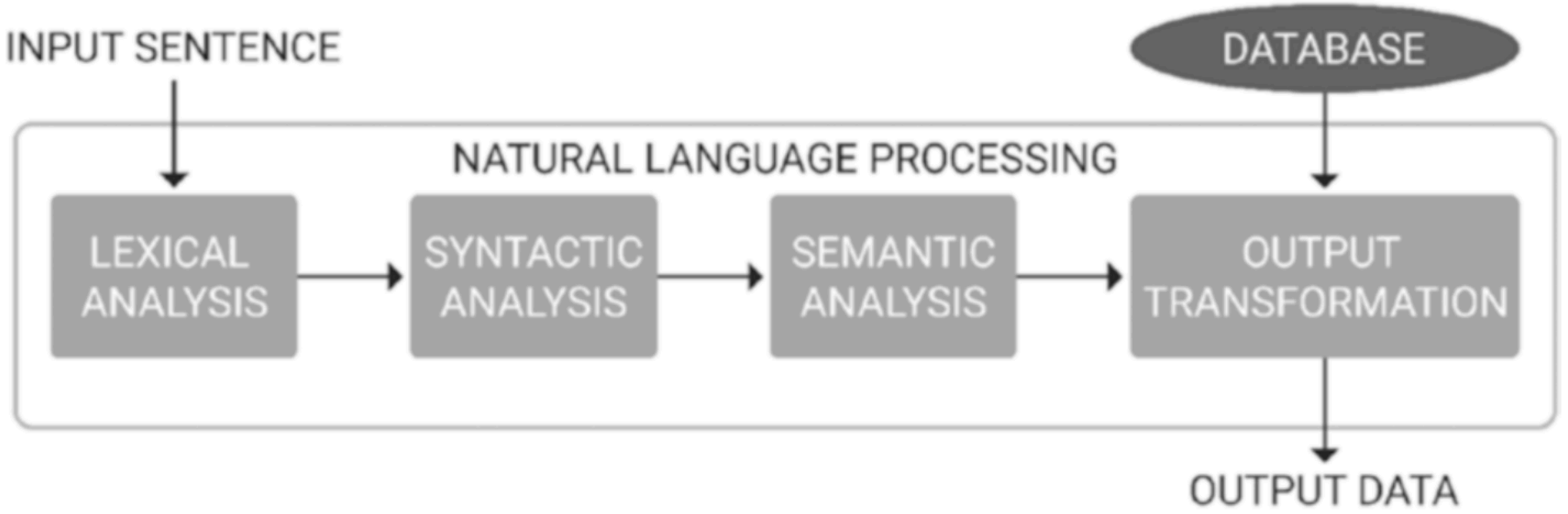
Machine Learning and AI for Healthcare

By Arjun Panesar

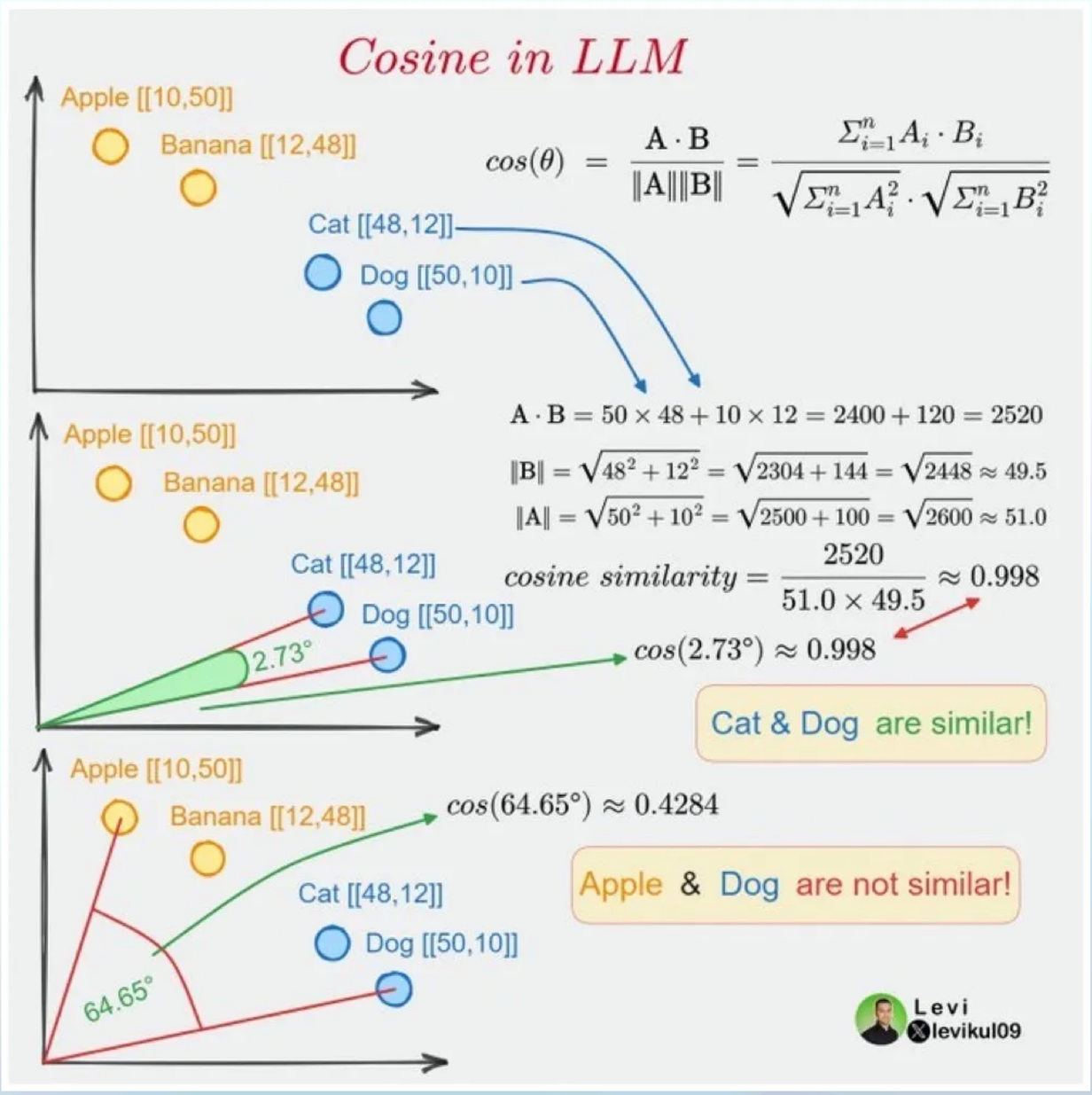
Dr. H. Mosavari


INPUT SENTENCE
DATABASE
NATURAL LANGUAGE PROCESSING
LEXICAL ANALYSIS
SYNTACTIC ANALYSIS
SEMANTIC ANALYSIS
OUTPUT TRANSFORMATION
OUTPUT DATA

Cosine Similarity

Python: SciPy; Method: cosine

- The cosine similarity is a metric that measures the similarity between two vectors.
- Simply put, and in the context of NLP — it's a measure of how similar the ideas and concepts represented in two pieces of text are.



Word embedding

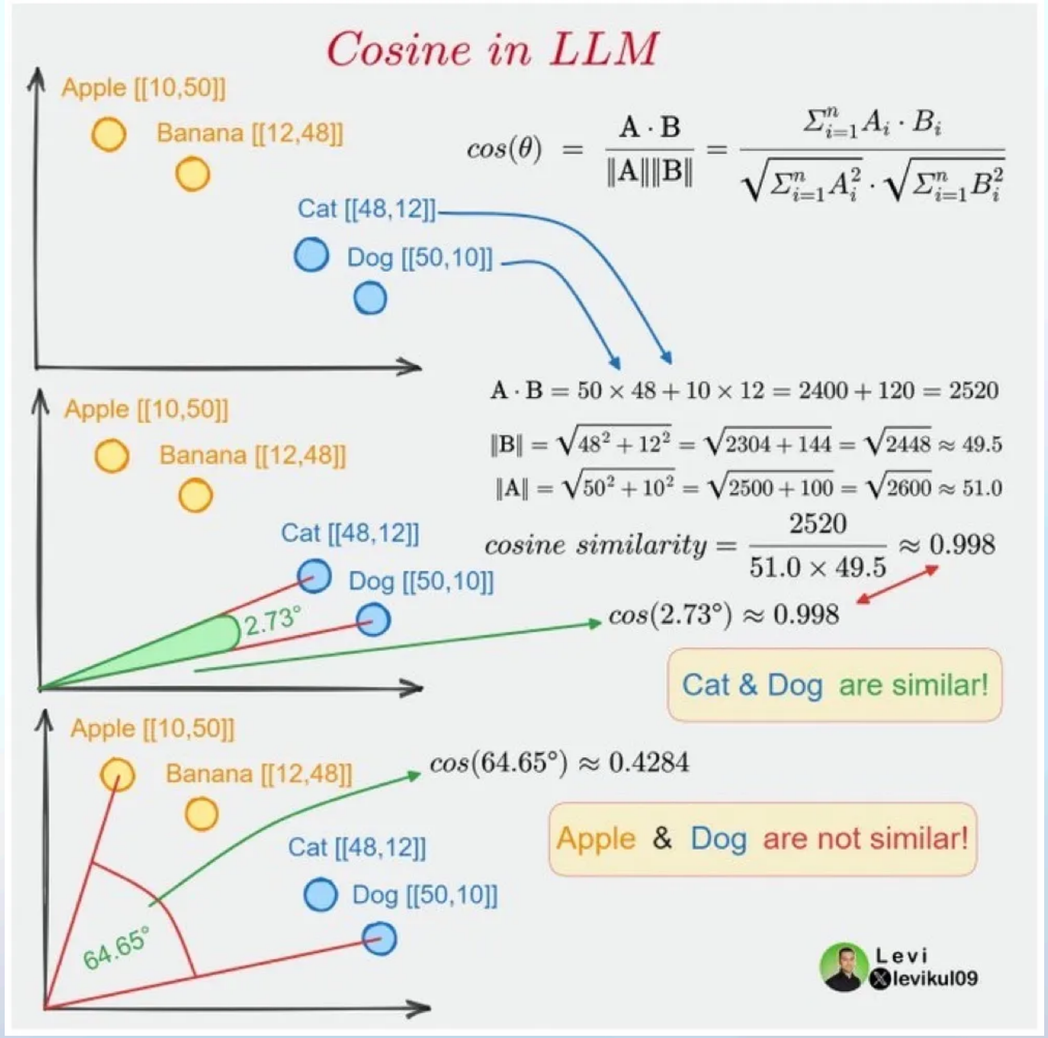
Words as Vectors

- Word embeddings are essentially vectors that capture the semantic essence of words.
- Imagine you have a vast library, and each book in that library is about a single word. The ‘location’ of each book in this imaginary library is determined by its content, or meaning. Words that are semantically similar would be located near each other. In computational terms, this ‘location’ is what a word embedding captures.
- Word embeddings are generated through neural networks trained on large text corpora. The network learns the contextual usage of words, effectively capturing not just the obvious synonyms but also the nuanced relationships between words.
- For example, the embeddings for “bank” in the context of a river and “bank” in a financial context would be different, even though the word is the same.

Back to cosine similarity

Cos (a)

- Cosine Similarity measures the cosine of the angle between two vectors. If two vectors are pointing in the same direction, the angle between them is zero, and the cosine is 1. If they are orthogonal, meaning they share no ‘directionality,’ the cosine is zero.
- In the context of word embeddings, think of each vector as an arrow pointing from the origin to the ‘location’ of a word in our imaginary library. Words that are semantically similar will have vectors pointing in similar directions, resulting in a higher Cosine Similarity.



Uses of Cosine similarity

How similar?

- Similarity between two documents
- Similarity between quesry (q) and document (d)
- Practically, however, calculating Sim(q,d) would prove computationally expensive as the number of documents used grows.

Naïve Bayesian Classifier

Naive but powerful

- **Key Idea:** It uses Bayes' Theorem, a mathematical rule for calculating probabilities, to predict the category (or class) of something based on prior knowledge and evidence.
- **Why “Naïve"?** It's called "naïve" because it assumes all features (inputs) are independent of each other, even though that's rarely true in real-world scenarios. Despite this oversimplification, it often works surprisingly well.

Naïve Bayesian Classifier

Naive but powerful

- **How Does It Work?**
  - **Learn** from data: During training, it calculates probabilities of each class and the likelihood of features within each class using the provided labeled data.
    - For example: "How often does the word 'free' appear in spam emails?"
  - **Predict** for new data: When given new data (e.g., an email), it combines:
    - The likelihood of seeing the features (e.g., words in the email) for each class.
    - The overall probability of each class (e.g., how common spam and non-spam emails are).
- It calculates which class (e.g., spam or not spam) is the most probable.

Naïve Bayesian Classifier

Naive but powerful

- **An Example:** Imagine you’re classifying emails as spam or not spam:
- Training:
  - You analyze many emails to learn:
  - Spam emails often contain "free," "offer," or "buy now."
  - Non-spam emails often contain "meeting," "agenda," or "project."
  - You also note that 30% of emails are spam, and 70% are not spam.
- Prediction:
  - A new email comes in: "Free meeting invite."
  - The classifier calculates:
  - Probability it's spam, given "free" and "meeting."
  - Probability it's not spam, given "free" and "meeting."
- It predicts the label with the higher probability.

Naïve Bayesian Classifier

Naive but powerful

- **Strengths**:
  - Simple and fast.
  - Works well for text classification (e.g., spam filters, sentiment analysis).
- **Limitations**:
  - Assumes feature independence (which may not be true).
  - Struggles when features have complex dependencies.

Genetic Algorithms

Learn from nature

- Evolution is considered the optimal learning algorithm.
- A Genetic Algorithm (GA) is a search and optimization technique inspired by the way nature evolves species over time. It mimics natural selection, where the best traits are passed down to future generations to solve problems or find optimal solutions.
- **Key Idea:** Think of it as a game where we:
    - Start with a bunch of possible solutions (like a population of creatures).
    - Combine the best solutions and make small changes (like breeding and mutations).
    - Repeat until we find the best solution (or a good enough one).

Genetic Algorithms

Learn from nature

- How Does It Work?
  - **Initialization:** Create a random population of possible solutions (called “individuals"). Each individual represents a potential answer, encoded as a string of "genes" (e.g., numbers or binary values).
  - **Fitness Evaluation:** Measure how "good" each individual is at solving the problem. This is called its fitness score. For example: If the problem is to find the shortest route between cities, the fitness score could be how short the route is.
  - **Selection**: Pick the "fittest" individuals (the best solutions so far) to create the next generation. Think of this as "survival of the fittest."
  - **Crossover (Recombination)**: Combine parts of two individuals (parents) to create new ones (offspring). Example: If Parent 1 = "1010" and Parent 2 = "0101," the child might be "1011."
  - **Mutation**: Make small random changes to some individuals to introduce variety and avoid getting stuck in bad solutions. Example: Change one "gene" in "1010" to get "1110."
  - **Repeat**

Genetic Algorithms

Learn from nature

- **An Example:**
- Let’s say we want to find the maximum value of a function f(x) = x^2, where x is between 0 and 31.
  - Population: Start with random binary strings (e.g., "10101" = 21, "00110" = 6).
  - Fitness: Evaluate f(x) for each string.
  - Selection: Keep the ones with the highest f(x).
  - Crossover: Combine two strings, like "10101" and "00110," to make "10110."
  - Mutation: Randomly flip a bit, e.g., "10110" → "11110."
  - Repeat until you find the highest f(x), which happens when x = 31

Genetic Algorithms

Learn from nature

- **GA have varied applications:**
  - Detection of blood vessels is ophthalmology imaging
  - Detecting the structure of RNA
  - Financial modeling
  - Routing vehicles

Best Practices and Considerations

Research goes slowley

Best Practices and Considerations

(1) Handle data well

- Mastering the art of machine learning takes time and experience. However, there are several areas worthy of consideration, particularly for beginners to machine learning, to ensure the best use of time and optimal model efficiency.
- **Good Data Management:** processes, policies, procedures, permissions, and certifications
- **Establish a Performance Baseline:** As there is no one perfect algorithm for every problem, attempt many algorithms to identify the relative performance of your models.
- **Spend Time Cleaning Your Data**

Best Practices and Considerations

(2) Modelling

- **Training Time:** If the dimensionality of data is large and computing power limited, training time will be extensive. For example, neural networks may not be appropriate for time-limited tasks.

- **Choosing an Appropriate Model:** Evaluating a range of models is a useful approach to machine learning. Occam's Razor is typically applied to choosing the model of choice.

Best Practices and Considerations

(3) Feature selection

- **Choosing Appropriate Variables:** Although more data is ordinarily invited in machine learning problems, it is typically preferable to work with fewer predictor variables for many reasons.

- **Redundancy:** ncreasing the number of variables within a training dataset increases the chances of models learning hidden relationships between them. It is vital to identify unnecessary variables and only use nonredundant predictor variables within models. A model that is learning redundant connections will impact model accuracy.

- **Overfitting**: Even if there are no relationships among predictor variables within a model, it is still beneficial to use fewer variables. Complex models, or those that use a high number of predictor variables, typically suffer from overfitting. As a result, models perform well on training datasets but are less accurate on validation and real-world settings, as they learn the error within the data (i.e., the noise) rather than the signal, or relationships, between variables.

Best Practices and Considerations

(3) feature selection

- **Productivity:** Practical considerations include the amount of data available, the subsequent effect on storage, computing resources, associated costs, time allocated for the project, and the time required for learning and validation.

- **Understandability:** Models with fewer predictor variables are easier to visualize, understand, and explain.

- **Accuracy**: Any machine learning model aims to generalize well. Depending on the use case, an approximation may be more beneficial than a precise, accurate output.

Best Practices and Considerations

(4) Mistakes with different weights

- **Impact of False Negatives:** When evaluating the impact of a model before deployment into a live environment, consider the impact of false negatives. For example, take a predictive model that classifies the risk of breast cancer. A false positive would mean that a patient is informed they have breast cancer when they do not, which should be identified later in the treatment pathway. However, a false negative would mean a patient with breast cancer would not be notified—which is potentially far worse and costly.

Best Practices and Considerations

(5) Optimization

- **Linearity:** Many machine learning algorithms assume that relationships are linear: in other words, that classes can be separated by a straight line of best fit or its higher dimensional representation. Data with a nonlinear trend may require transforming to make the relationship linear: for instance, log-transforming data where there is an exponential relationship.

- **Parameters:** Each machine learning model is subject to parameters and hyperparameters.

- **Ensembles:** In specific machine learning problems, it may prove more useful to group classifiers together using the techniques of voting, weighting, and combination to identify the most accurate classifier possible. Ensemble learners are very useful in this aspect.