SkillNavora · Written by SkillNavora · Plain-English guide · Fake data only

NLP, Neural Networks and LPUs, Explained Simply

Three words come up again and again in AI: **natural language processing (NLP)**, **neural networks** and **LPU**. This guide explains each one in plain English, shows how they fit together, and links to the deeper lessons on this site. You do not need any background.

> **The short version.** A **neural network** is a program that learns patterns from examples. **NLP** is the job of making computers work with human language. An **LPU** is a kind of chip built to run language models quickly. Most modern language AI uses all three ideas together.

Part 1. Neural networks in one picture

A neural network is built from many small units called **neurons**. Each neuron takes numbers in, multiplies each one by a **weight**, adds them up, and passes the result on. Learning means adjusting the weights until the network's answers match the examples.

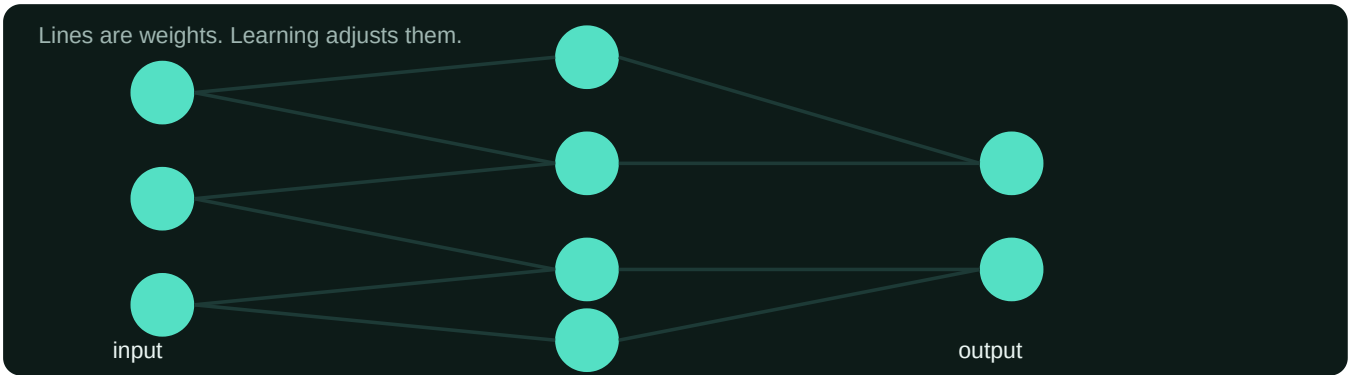


Figure 1. Input, hidden layer and output. Each line carries a number that learning adjusts.

Three words to know

Word	Plain meaning
Weight	How strongly one neuron's signal affects the next
Layer	A group of neurons that work at the same stage
Training	Adjusting weights many times until answers improve

Deep learning just means a neural network with many layers. Lesson ML-26 on this site builds one step by step.

Part 2. Natural language processing: teaching computers to read

Human language is messy. The same idea can be written many ways, and meaning depends on context. NLP is the set of methods that turn text into something a computer can use.

NLP task	What it does	Example
Cleaning	Removes noise such as extra spaces and symbols	"Great!!! product" becomes "great product"
Tokenising	Splits text into pieces (tokens)	"I love it" becomes 3 tokens
Sentiment	Decides if text is positive or negative	"The app crashes" becomes negative
Classification	Puts text into groups	A complaint goes to "billing"
Summarising	Shortens text and keeps the key points	A 10-page policy becomes 5 bullets
Question answering	Finds or writes an answer from text	"How many leave days?" answered from a policy

From words to numbers

A computer cannot use words directly. It turns each token into a list of numbers called an **embedding**. Words with similar meaning get similar numbers. Earlier methods counted words (for example TF-IDF, which gives more weight to words that are rare across documents but common in one). Modern language models learn their embeddings from very large amounts of text.

Lesson ML-25 on this site walks through cleaning, TF-IDF and sentiment classification with working code.

Part 3. How NLP and neural networks meet

Most modern NLP uses neural networks. A language model is a large neural network that predicts the next token. It learns this by reading very large amounts of text and adjusting its weights each time it guesses wrong.

1. **Input:** text is split into tokens and turned into embeddings.
2. **Processing:** many layers combine the embeddings, so each token takes context from the others.
3. **Output:** a score for every possible next token. The model picks one (see the tokens guide on sampling).
4. **Repeat:** the chosen token is added to the input, and the process runs again for the next token.

Part 4. What an LPU is

An **LPU** (language processing unit) is a type of processor designed to run language models quickly. Groq is one company that uses the name for its chips. Most AI today runs on **GPUs** (graphics processors), which are good at many kinds of parallel maths. An LPU is designed around the sequential way language models generate tokens one after another.

Word	What it is	Why you care
CPU	General-purpose processor in every computer	Runs everyday software; slow for large models
GPU	Processor with thousands of small cores	The most common hardware for training and running AI
LPU	Chip designed for fast language model output (for example Groq's)	Some providers offer very fast responses on it

> **Honesty note.** Chip speeds, availability and prices change quickly and are often quoted by the company that sells them. Check the provider's current documentation before you rely on any speed or cost claim. This guide does not state any LPU performance figure.

Why it matters for a product manager: faster responses can change what users will accept. Speed, cost and quality trade off against each other, so the choice of hardware and model is a product decision, not only an engineering one.

Part 5. A quick map of the terms

Term	Layer	One-line meaning
NLP	Goal	Making computers work with human language
Neural network	Method	Learning patterns with weighted layers of neurons
Language model	Product	A neural network trained to predict and write text
LPU / GPU / CPU	Hardware	The chip that runs the model

Part 6. Worked example: a support-ticket sorter

1. **Goal:** sort fake support tickets into Billing, Bug or Other.
2. **Classic NLP:** clean the text, turn it into TF-IDF numbers, train a simple classifier on 50 labelled fake tickets. Measure accuracy on 20 held-out tickets.

3. **Language model:** ask the model to choose one label from the list and give a one-line reason. Lock the output to the three labels.
4. **Compare:** speed, cost per ticket (see the tokens guide), and accuracy on the same 20 tickets.
5. **Choose:** the simplest method that meets your accuracy target. Use a language model where the rules are hard to write.

Pass when: you can explain which method you chose, what it costs per ticket, and what accuracy it reached on the same test set.

Part 7. Common mistakes

Mistake	Better habit
Using a large model where a simple rule works	Start with the simplest method; measure before you upgrade
Judging speed from a marketing claim	Time the same 20 requests yourself
Confusing the model with the chip	Name both separately in every plan
Testing only on clean sentences	Include typos, slang and mixed languages

Part 8. Check yourself

1. What does a weight do in a neural network?
2. Name one NLP task that is not a language model.
3. What is an embedding?
4. What does LPU stand for, and what is it designed for?
5. Why should you test speed yourself?

Answers: 1. It sets how strongly one signal affects the next; learning adjusts it. 2. Sentiment analysis, classification or cleaning. 3. A list of numbers that represents the meaning of a token. 4. Language processing unit; it is designed to run language models quickly. 5. Published claims change and may not match your use.

Part 9. Glossary

Term	Plain meaning
NLP	Natural language processing: computers working with human language.
Neural network	A program that learns patterns through weighted layers.

Weight	A number learning adjusts to change a network's output.
Embedding	Numbers that stand for the meaning of a token.
TF-IDF	A word-weighting method for documents, used in classic NLP.
LPU	Language processing unit: a chip designed for fast language model output.
GPU	Graphics processor, widely used to run and train AI models.

Related lessons on SkillNavora: ML-25 (NLP and text analytics), ML-26 (neural networks and deep learning), GenAI-01 (how large language models work).

Part 10. The NLP pipeline, end to end

Every text system follows the same stages. Knowing the stages tells you where a problem starts when an answer is wrong.

Stage	What happens	Typical failure
1. Collect	Gather the text (tickets, reviews, documents)	Missing or duplicated records
2. Clean	Remove noise: extra spaces, markup, repeated symbols	Removing useful signs such as a rupee symbol or a question mark
3. Tokenise	Split text into tokens (words or word pieces)	Names and mixed languages split in odd places
4. Represent	Turn tokens into numbers (counts, TF-IDF or embeddings)	Rare terms get too much or too little weight
5. Model	Learn a rule or pattern from labelled examples	Too little training data; labels inconsistent
6. Evaluate	Measure on examples the model never saw	Testing on the same examples used for training
7. Serve	Run the model on new text in a product	Slow responses; unexpected input types
8. Monitor	Track quality and drift over time	Quality drops quietly as language changes

> **Debugging rule:** when an answer is wrong, check the stages in order. Most problems start at stage 2, 4 or 5, not at the model.

Part 11. Classic NLP methods, with numbers

11.1 Bag of words

A bag of words ignores order and counts how often each word appears. It is simple, fast and often good enough for sorting tickets.

Document	cheap	flight	delay	hotel
D1: "cheap flight cheap"	2	1	0	0
D2: "flight delay"	0	1	1	0
D3: "hotel cheap"	1	0	0	1

11.2 TF-IDF, worked by hand

TF-IDF gives more weight to words that appear often in one document but rarely across all documents. Here is the calculation for the word *cheap* in D1.

1. **Term frequency (TF):** count of the word in the document ÷ total words in the document. D1 has 3 words, and *cheap* appears 2 times. TF = 2 ÷ 3 = 0.667.
2. **Document frequency:** *cheap* appears in 2 of 3 documents.
3. **Inverse document frequency (IDF):** ln(3 ÷ 2) = 0.405.
4. **TF-IDF:** 0.667 × 0.405 = **0.270**.

For *flight* in D1: TF = 1 ÷ 3 = 0.333; it also appears in 2 documents, so IDF = 0.405; TF-IDF = **0.135**. *Cheap* scores higher in D1 because it appears twice there, which is the behaviour we want.

11.3 Lexicon sentiment, the simplest version

A lexicon is a list of words with scores. Add the scores in a sentence to get a sentiment.

Word	Score
love	+2
good	+1
crash	−2
slow	−1

"The app is slow and crashes" scores −1 + (−2) = −3, so it is negative. The method fails on sarcasm and on words the list does not know, which is why labelled data and learned models exist.

Part 12. How do you know a classifier is good?

Accuracy alone can mislead. Use a confusion table on held-out examples: 20 fake tickets, 10 real "bug" and 10 real "not bug".

	Actually a bug	Actually not a bug
Predicted bug	6 (true positives)	2 (false positives)
Predicted not bug	4 (false negatives)	8 (true negatives)

Metric	Formula	Value	Plain meaning
Accuracy	(6 + 8) ÷ 20	0.70	How often the label is right overall
Precision	6 ÷ 8	0.75	When it says bug, how often it is right
Recall	6 ÷ 10	0.60	Of the real bugs, how many it found
F1 score	2 × 0.75 × 0.60 ÷ (0.75 + 0.60)	0.67	One number that balances precision and recall

> **Choose the metric from the cost of mistakes.** For a safety report, missing a real problem (low recall) is worse. For a refund approval, a wrong yes (low precision) is worse.

Part 13. Training a neural network: one step, by hand

Learning is repeated small corrections. This tiny example has one weight, one input and one target. It shows the four moves that every neural network makes.

1. **Predict:** input x = 1, weight w = 0.5. Prediction = w × x = 0.5.
2. **Measure the error (loss):** target = 2. Loss = (prediction − target)² = (0.5 − 2)² = 2.25.
3. **Find the direction to change the weight (gradient):** 2 × (prediction − target) × x = 2 × (−1.5) × 1 = −3.
4. **Update:** learning rate 0.1. New w = 0.5 − 0.1 × (−3) = **0.8**.

Run it again with w = 0.8: prediction 0.8, loss (0.8 − 2)² = 1.44. The loss fell from 2.25 to 1.44 in one step. A real network does this for millions of weights at once, over millions of examples.

Word	Plain meaning
Loss	A score for how wrong the prediction is; lower is better
Gradient	The direction that makes the loss smaller
Learning rate	How big each update step is; too big overshoots, too small is slow

Epoch	One full pass through the training examples
Overfitting	Learning the training examples by heart, and failing on new ones

Part 14. Embeddings and similarity, with numbers

An embedding is a list of numbers for a word or sentence. Similar meanings point in similar directions. Cosine similarity measures the angle between two lists: 1 means the same direction, 0 means unrelated.

Pair	Vectors	Cosine similarity
"annual leave" and "holiday rules"	[0.9, 0.2] and [0.85, 0.25]	about 0.998 (very similar)
"annual leave" and "dress code"	[0.9, 0.2] and [0.1, 0.9]	about 0.32 (weakly related)

The numbers are toy values. Real embeddings have hundreds of dimensions, but the comparison works the same way.

Part 15. From a neural network to a language model

1. **Input:** the text so far, as tokens and embeddings (see the tokens guide).
2. **Layers:** many neural layers combine information, and attention lets each token take context from earlier tokens.
3. **Scores:** one score for every possible next token. Example: scores 2.0, 1.0 and 0.0 become probabilities 0.665, 0.245 and 0.090 (softmax).
4. **Choose:** sampling picks one token, using settings such as temperature.
5. **Repeat:** the chosen token is added to the input and the cycle runs again until the answer is complete.

The model was trained by predicting the next token in very large amounts of text and adjusting its weights each time it was wrong, which is the same loop as Part 13 at a much larger scale.

Part 16. Training and inference: two different jobs

	Training	Inference (running the model)
Purpose	Learn the weights	Produce an answer for a user
Frequency	Rare, large, expensive	Constant, every request
Who usually does it	Model builders	Product teams using a provider

What you watch	Quality on test data	Speed, cost per request, errors

As a product manager, most of your decisions are about inference: which model, how fast, at what cost, and with what checks.

Part 17. Where LPUs, GPUs and CPUs fit

Chip	General idea	Where you meet it
CPU	A few powerful cores; good for varied everyday tasks	Your laptop and servers for ordinary software
GPU	Thousands of small cores; good for many calculations at once	Most AI training and much AI serving
LPU	A chip designed for fast language model output (Groq uses the name)	Some providers offer LPU-backed models

> **Honesty note.** Do not quote speed numbers from marketing. Test the same prompt on each provider yourself, measure time to first token and total time, and check the provider's current documentation for specifications and availability.

17.1 A fair speed test

1. Pick 20 fake prompts of the length your product uses.
2. Run each prompt on each option, at the same time of day, three times.
3. Record time to first token and total time. Use the median, not the best run.
4. Record the cost per request from the usage numbers (tokens guide).
5. **Pass when:** you can state the median speed and cost per option, with the date and the test conditions.

Part 18. Bias, errors and limits

Risk	Example	Control
Biased training data	A classifier learns that one dialect is "rude"	Test on every group of users; review errors by group
Unseen words	A new product name breaks the sorter	Keep a list of unknown terms and review it weekly
Confident wrong answers	A language model states a wrong policy	Ground answers in documents; allow "I don't know"

Sensitive text	Personal data in a support ticket	Remove personal data before storage or sending
Drift	New slang lowers accuracy over months	Re-test monthly on fresh labelled examples

Part 19. Choosing the approach: a decision guide

1. **Is the rule simple and fixed?** Use a rule or a keyword list. Stop here if it works.
2. **Do you have labelled examples?** If not, label 200 examples first. A model cannot learn what you never showed it.
3. **Is the classic model accurate enough?** Use TF-IDF with a simple classifier. It is cheap and easy to explain.
4. **Does the task need understanding of meaning or long context?** Use a language model, grounded in your documents (see the RAG guide).
5. **Is speed critical?** Measure the options with the test in Part 17.
6. **Is the output used for a decision about a person?** A person must review it. Automation prepares, a human decides.

Part 20. Hands-on labs (fake data only)

1. **Tokenise and count.** Take ten fake sentences. Count words per sentence and list the five most common. Pass when you can explain why "the" is common and not useful.
2. **TF-IDF by hand.** Repeat Part 11 with three new fake documents. Pass when your numbers match a calculator.
3. **Confusion table.** Build a table for twenty fake labels with a deliberately weak rule. Pass when you can name the metric that matters for your use.
4. **One training step.** Change the starting weight and learning rate in Part 13. Pass when you can predict whether the loss falls.
5. **Similarity.** Make three fake embedding pairs and compute cosine similarity. Pass when the closest pair matches your intuition about meaning.
6. **Fair speed test.** Run Part 17 on two options with five prompts. Pass when you have a median and a cost per request.

Part 21. Product manager questions

Question	What a good answer includes
Why not always use the biggest model?	Cost and speed grow with size; a smaller model may pass the test

What accuracy is enough?	The one that keeps the cost of the worst mistake acceptable, measured on your data
What happens when the model is wrong?	A fallback path, a human review, and a logged error
How will we know quality drops?	A monthly test set and an alert threshold
Who owns the decision?	A named person for every outcome that affects a customer

Part 22. Extended quiz (answers follow)

1. In TF-IDF, what does the IDF part reduce the weight of?
2. Why is accuracy alone often misleading?
3. In the one-step example, why did the weight go from 0.5 to 0.8?
4. What is the difference between training and inference?
5. When would you use recall over precision?
6. Why measure speed yourself?

Answers: 1. Words that appear in many documents, which are less useful for telling documents apart. 2. A model can score well by always predicting the common label when classes are unequal. 3. The gradient was negative, so increasing the weight reduced the loss. 4. Training learns the weights once; inference runs the model on each request. 5. When missing a real case is costly, such as a safety issue. 6. Published speeds may not match your prompts, region or load.

Part 23. Glossary (extended)

Term	Plain meaning
Corpus	The collection of text used for analysis or training
Stop words	Very common words, such as "the", often removed for classic methods
Lemmatisation	Reducing a word to its base form, such as "running" to "run"
Confusion table	Counts of right and wrong predictions by class
Held-out data	Examples kept aside to test the model
Softmax	A function that turns scores into probabilities that add up to 1
Inference	Running a trained model on new input
Drift	A gradual change in input that lowers model quality

Part 24. One-page cheat sheet

Need	Start with
Simple, fixed rule	Keyword list
Sort text into groups, with labels	TF-IDF and a simple classifier
Understand meaning or long context	Embeddings and a language model, grounded in documents
Measure quality	Confusion table, precision, recall, F1 on held-out data
Measure speed and cost	The fair speed test, median not best
Keep quality high	Monthly re-test and a human review path

Part 25. Final checklist

Item	Done
I can describe each NLP pipeline stage and where failures start	[ ]
I can compute TF-IDF for one word by hand	[ ]
I can explain precision, recall and F1 with my own numbers	[ ]
I can explain one training step in plain words	[ ]
I ran a fair speed test before quoting any speed	[ ]
I named a human owner for every decision about a person	[ ]

Part 26. Inside a neural network: layers and activation

A layer multiplies its inputs by weights, adds a bias (a number that shifts the result), and passes the sum through an **activation function**. The activation decides how strongly a neuron fires. Without it, many layers would collapse into one simple calculation.

Activation	Rule	Example input → output	Used for
ReLU	Negative becomes 0; positive stays	−2 → 0; 3 → 3	Hidden layers in many networks
Sigmoid	Squashes any number into 0 to 1	0 → 0.5; 2 → 0.88	A yes-or-no output probability

Softmax	Turns a list of scores into probabilities that add to 1	[2, 1, 0] → [0.665, 0.245, 0.090]	Choosing one of many labels or tokens

26.1 A tiny forward pass

1. **Inputs:** two numbers, [1, 2].
2. **Weights for one neuron:** [0.5, −0.25]; bias 0.1.
3. **Weighted sum:** 1 × 0.5 + 2 × (−0.25) + 0.1 = 0.1.
4. **ReLU:** 0.1 stays 0.1, so the neuron outputs 0.1.

A real network stacks thousands of these neurons in many layers. Training adjusts every weight and bias using the loss and gradient from Part 13.

Part 27. Train, validation and test: keeping the score honest

Split	Purpose	Typical share (example)
Training	The model learns weights from it	70 percent
Validation	You tune settings (for example, learning rate) on it	15 percent
Test	You report the final score on it, once	15 percent

Overfitting in plain words: training accuracy rises while validation accuracy falls. The model memorised the examples. Fixes include more varied data, fewer parameters, early stopping (stop when validation stops improving) and regularisation (a penalty for very large weights).

> **Golden rule:** never tune on the test set. Once you look at it and change something, it is no longer a fair test.

Part 28. A full project: support ticket sorter (fake data)

28.1 Plan on one page

Item	Answer
Goal	Sort fake support tickets into Billing, Bug or Other
Data	200 fake tickets, labelled by you; no real customer text
Success	Macro F1 of at least 0.80 on the test set, and no bug missed in the urgent group
Must never	Route a ticket to a person without a fallback; store personal data

28.2 Build in seven steps

1. **Label 200 fake tickets** with one label each. Write the labelling rule on one page first, and check that two people would agree on 20 of them.
2. **Split** 70 percent training, 15 percent validation, 15 percent test (Part 27). Put the test set in a locked file you do not open until the end.
3. **Baseline with keywords:** a list of words for each label. Record the score; this is the bar a model must beat.
4. **Classic model:** TF-IDF and a simple classifier, tuned on validation only.
5. **Language model option:** one fixed-label prompt, tested on the same validation set. Compare cost per ticket (tokens guide).
6. **Final test:** run the chosen method once on the test set. Report accuracy, precision, recall and F1 per label.
7. **Failure review:** list ten wrong predictions and name the cause of each (unclear ticket, missing word, label disagreement).

28.3 What the report must contain

Section	Content
Data	How many tickets, how labelled, the agreement check
Method	Keyword baseline, classic model and language model, with settings
Result	One table with the four metrics for each option, on the test set
Cost and speed	Median time and cost per ticket for the two learned options
Failures	Ten examples and their causes
Decision	Chosen method, why, and the fallback for uncertain tickets

Part 29. Common project mistakes

Mistake	Consequence	Fix
Tuning on the test set	An inflated score you cannot trust	Keep the test set locked until the final run
Inconsistent labels	The model learns noise	Write the rule; check agreement on 20 examples
Reporting accuracy only	A rare but important class is missed	Report precision and recall per label
Quoting a vendor speed	Wrong expectations for your users	Run the fair speed test

No fallback	Uncertain tickets are dropped	Send low-confidence or failed cases to a person

Part 30. Inference cost, without vendor prices

For any learned option, cost per request has the same structure. Plug in your own current rates.

Cost per request = (input tokens ÷ 1,000,000 × input rate) + (output tokens ÷ 1,000,000 × output rate)
Monthly cost = cost per request × requests per month
Latency = time to first token + time to finish the answer (use the median of 20 runs)

Choice	Cost driver	Speed driver
Keyword rule	Almost nothing per request	Very fast
TF-IDF with a classifier	Low; runs on your own server	Fast
Language model	Tokens per request; grows with long input	Depends on model size and hardware

Part 31. Explain it to a non-technical stakeholder

1. **Neural network:** "A system that learns by making small corrections after each mistake, using many examples."
2. **NLP:** "Teaching software to sort, search and summarise written language."
3. **Embedding:** "A numeric fingerprint of meaning, so similar texts can be compared."
4. **Precision and recall:** "How often a flag is right, and how many real cases the flags catch."
5. **LPU:** "A chip designed to produce language model answers quickly. We test any speed claim before we trust it."

Part 32. Extended practice quiz (answers follow)

1. Why does the activation function matter?
2. What is the purpose of the validation set?
3. Training accuracy rises but test accuracy falls. What is happening?
4. In the project, what must never be tuned?
5. Why report cost per request and latency together?

Answers: 1. Without it, stacked layers collapse into one simple calculation, so the network cannot learn complex patterns. 2. To tune settings without touching the test set. 3. Overfitting: the model memorised training examples. 4. The test set. 5. Speed and cost trade off; a fast cheap option may be less accurate.

Part 33. Reference card

Term	Plain meaning
Bias term	A number added to a weighted sum that shifts the result
ReLU, sigmoid, softmax	Activation functions that shape neuron outputs
Validation set	Examples used to tune settings
Early stopping	Stopping training when validation stops improving
Regularisation	A penalty that keeps weights small, to reduce overfitting
Macro F1	The average F1 across labels, giving each label equal weight
Time to first token	How long until the first word of an answer appears