SANSKRIT AND AI / PART 01 / STUDY NOTES

Sanskrit and AI, Part 1: Pāṇini's Machine

Study notes by Sanskrit AI Universe, based on this video

Video by Dr P.R.K. Varma, Founding Editor · DrVARMA PRK · 14 Nov 2026

Not the creator's own slides

About these notes

— Official title: Sanskrit and AI part 1 : Pāṇini's Machine | Dr.P.R.K.Varma.,M.D.,D.M.

— Speaker and channel: Dr P.R.K. Varma, Founding Editor · DrVARMA PRK

— Published 14 Nov 2026 · Running time 14 min 39 s · https://www.youtube.com/watch?v=tBn1tvgUzgA

— These pages are study notes drawn from the video's own title, description and captions, plus the journal's already published Watch note. They are not a transcript and not a substitute for watching the film.

— The journal treats videos as leads, not as sources you should cite on their own. Check the papers and editions behind a claim.

— Caption source: English auto-generated subtitles. Wording may be imperfect. Distinctive phrases follow the captions closely.

This part in brief

— There are two ways to capture a language. One is to write its rules.

— 25 centuries ago, a grammarian named Pāṇini described all of Sanskrit in fewer than 4,000 short rules.

— The other is to train a machine. Today, a large language model reads trillions of words and adjusts billions of numbers until language emerges.

Chapter one, the grammar machine

— Pāṇini's grammar is called the Aṣṭādhyāyī, the eight chapters.

— Eight books, each in four sections, hold nearly 4,000 sūtras.

— Each sūtra is only a few syllables long. Around the rules sit three data files, a table of sounds, a list of about 2,000 verbal roots and lists of word groups.

Principle one, compression

- The grammar opens with 14 short strings of sounds, the Śiva Sūtras.
- Pick any sound, add a letter marker, and you have named every sound in between.
- The two-letter code ach means all the vowels. The code hal means all the consonants.

Principle two, metadata

- Pāṇini attaches silent tag to roots and suffixes. But before they vanish, they tell the grammar what to do.
- Take the root kri to make a suffix tagged with k protects the vowel.
- A suffix tagged with n strengthens it fully.

Principle three, a formal language for writing rules

- Look at one famous rule, three words Pāṇini turns the grammatical cases into operators.
- The locative is the context on the right. The ablative is the context on the left.
- So the rule reads i, u, ri and lri become ya, va, ra and la before a vowel.

Principle four, zero and inheritance

- Pāṇini gives absence a name, lopa not appearing. An element can be deleted and still do its work.
- In the word agnicit the suffix vanishes completely, yet it still adds the final t.
- And when one element replaces another, the substitute the substitute inherits the properties of the original.

Principle five, context that carries forward

- A sūtra almost never repeats itself. Words from earlier rules flow down into later ones.
- This is called Heading rules govern hundreds of rules beneath them exactly like scope in a program.
- So no rule can be read alone.

Principle six, conflict resolution

— When two rules can both apply, which one wins? wins?

— Second, for rules of equal strength, sūtra one for two decides.

— Tradition reads it as the later rule wins. A recent Cambridge thesis argues it means the rule acting on the right-hand side of the word wins.

Principle seven, meaning before word order

— Pāṇini defines six roles around every action. They are called the independent agent, the object most desired, the instrument, the recipient, the source, and the location.

— The roles are carried by word endings, not by position.

— Rama kills Ravana with an arrow.

Principle eight, rules that call data

- A rule simply names the first item and says etc., and the whole list is pulled in.
- A compact rule core with an external memory looked up on demand.
- We want to say he becomes start with the root and the present tense marker lie.
- The marker is replaced by the third person ending t.

Chapter two, the learning machine

- At the top, the model outputs a probability for every token in its vocabulary.
- If the model gave the true token a low probability, the loss is high.
- For every one of its billions of weights, the model computes which weight to nudge it to reduce the loss.
- Out comes a model that can continue any text, but it is not yet an assistant.

Chapter three, where the two machines meet

— One, tokenizers that cut words at real roots and suffixes, not at statistical accidents. statistical accidents.

— A Pāṇinian engine can generate millions of correct forms, each with its full derivation, precious for languages with little text.

— Three, training on the steps, not just the answers, so the model learns to derive, not to guess.

— Five, constrained decoding, where the rule engine filters every token the model proposes.

Sources and credit

— Study notes by Sanskrit AI Universe, based on the video “Sanskrit and AI part 1 : Pāṇini's Machine | Dr.P.R.K.Varma.,M.D.,D.M.”. These are not the creator's own slides.

— Video by Dr P.R.K. Varma, Founding Editor, posted on the YouTube channel DrVARMA PRK.

— Watch the video: https://www.youtube.com/watch?v=tBn1tvgUzgA

— Captions used: English (auto-generated) (en), retrieved on 24 November 2026 via a public Invidious instance. Auto-captions can mis-hear words; obvious proper-name spellings were given in IAST where the journal already uses them.

— Also used: the official video title and description, and the Watch catalogue note already published in this journal (presenter and why the video is listed).

— If the Founding Editor later supplies his own deck for this part, replace the PPTX and PDF in src_assets/slides/ (see the journal README, “How to swap in RK's own slides”).