# Doing an Undergraduate Thesis in Machine Learning or NLP

Finding a topic, doing the work, writing it up, defending it

**Konstantinos Skianis**

Assistant Professor of Natural Language Processing

Department of Computer Science and Engineering, University of Ioannina

y3nk0.github.io

October 2026 ▪ companion to the full guide and the one-page cheat sheet

# What a thesis is — and is not

A supervised project, 4–9 months, that shows you can take a research-level problem, work methodically, and communicate the result.

## What examiners look for

- A **clear question**, in one sentence.
- **Correct methodology** — sane baselines, honest evaluation, no leakage.
- **Understanding.** You must explain every design decision and every line you submit.
- **Clear writing** and **honest limitations**.

## Not required

Beating the state of the art. A new architecture. Training a large model from scratch.

A thesis that carefully reproduces, analyses and extends existing work is a *good* thesis. A record chase ending in one unexplained number is not.

## Scope: the single most common failure

Almost every thesis that goes badly goes badly the same way: the topic was too large, six months went into getting a pipeline to run, and there was nothing to say about it.

### The three-question test

1. Can you name the **dataset** and the **metric** today?
2. Can you run **one baseline** on your hardware (or free cloud GPUs) within **two weeks**?
3. If your main idea fails completely, is there *still* a thesis — a comparison, an analysis, an error study?

Three yeses: the scope is right. Any no: narrow it with your supervisor *before* you write code.

Part 1

# Finding a topic

## Step zero: check what is already on offer

1. **Look for advertised topics first.** Most supervisors keep a list: their web page, the department's thesis board, a shared document. Ask older students where it lives.
2. **Pick one and do a small piece of research before the first meeting:** the two or three papers it points to, the dataset, the metric, one existing implementation skimmed, three written questions.
3. **Only if you are given freedom**, or nothing on the list fits, use the three routes that follow.

### Why it matters

Arriving prepared turns the first meeting from an introduction into a plan, and a topic the supervisor already cares about comes with data, baselines and attention attached.

# You do not need an original idea

You need a **well-defined problem** with data and a way to measure success. There are three reliable routes to one.

| Route | What you do | Best if |
|---|---|---|
| **A. Shared task or challenge** | Enter an organised competition: the data, the metric and the deadline already exist | You want structure and a fast start |
| **B. Application thesis** | Take a method that works and apply it carefully to a new domain, language or setting | You like building systems |
| **C. Analysis or theory** | Reproduce, probe, compare, measure: explain *why* methods behave as they do | You like understanding over building |

# Route A — why a shared task fits so well

- **The hardest part is done for you:** problem definition, data, annotation and the evaluation protocol — the parts that sink inexperienced projects.
- **You cannot fool yourself.** A hidden test set is an honest measure.
- **Real baselines.** Organisers publish one; past editions give you a whole comparison table free.
- **A deadline** — the best project-management tool ever invented.
- **It can end in a publication.** System description papers are peer-reviewed, archival, and genuinely achievable — often your first paper, with your supervisor.
- **It is a portfolio item.** "4th of 31 teams" is concrete in a way that "did a thesis on sentiment analysis" is not.

## A shared task, defined

Organisers publish a precise problem, training and development data, an official metric and an evaluation window. Everyone is scored on the same hidden test set.

# Where to find them

- **SemEval** (semeval.github.io) — a dozen or more tasks a year: semantic similarity, sentiment and emotion, irony, hate speech, factuality and hallucination detection, clinical and legal NLP, multilingual tasks.
- **Workshop shared tasks at *ACL venues** — most workshops at ACL, EMNLP, NAACL, COLING and EACL run one: **WMT** (translation), **CoNLL**, **BabyLM**, **BioNLP**, **WNUT**, **CheckThat!** Check the ACL Anthology and each conference's workshop list.
- **CLEF labs** (clef-initiative.eu) and **TREC tracks** (trec.nist.gov) — retrieval, biomedical QA (**BioASQ**), argument mining, authorship, misinformation.
- **Competition tracks at ML conferences** — the NeurIPS competition track and similar programmes.
- **Platforms** — Kaggle, CodaBench, EvalAI, AIcrowd, DrivenData. Kaggle alone is usually not a thesis — you still need a question and an analysis chapter.
- **Leaderboard hubs** — Hugging Face hosts datasets, models and evaluations; many leaderboards stay open after the official window.

# Running a shared task as a thesis

1. **Shortlist two or three tasks** and read last year's *task overview paper* plus three or four system papers — 30 minutes each.
2. **Reproduce the organisers' baseline first.** If you cannot match it, nothing else is interpretable.
3. **Build your own validation split** and trust it; leaderboard submissions are limited.
4. **Log every run** — config, seed, score, what changed. The log becomes your results chapter.
5. **Submit something early**, even weak. Pipelines always break.
6. **Write the system paper**; the thesis is its long version.

## Check the dates before you commit

Registration → data release → development phase → a short evaluation window → system paper deadline. Dates change every edition. An evaluation window that opens after your submission date is a thesis you cannot finish.

## Window already closed?

Use it anyway. Past data, metrics and leaderboards are usually public: run it retrospectively and report where you *would* have ranked.

# Route B — an application thesis

## Where to point it

- **A language with few resources.** Most results are reported on English; Greek has far fewer evaluated baselines.
- **A domain with its own vocabulary:** legal, medical, financial, educational, archival text; code; logs.
- **A deployment constraint:** runs on a laptop or a phone, answers in under 100 ms, works with 50 labels, never leaves the building.
- **A real user with a real problem** — another department, a university service, a local company.

## What makes it rigorous, not a demo

- Two baselines, one embarrassingly simple. If the simple one wins, that *is* your finding.
- A split fixed *before* you look at results; the test set touched once.
- Several seeds, with mean and spread.
- An error analysis: 50–100 mistakes categorised by hand — the chapter examiners remember.
- A cost column: parameters, training time, latency, hardware.

# Route C — analysis or theory

You do not have to build something new to contribute. Explaining *why* existing methods behave as they do is harder, and often more valuable.

- **Reproduction and stress-testing** — other seeds, datasets, languages, hyperparameters.
- **Ablation and attribution** — five components, one number: which one actually matters? Often not the one in the title.
- **Robustness** — typos, dialect, code-switching, domain shift, longer inputs, class imbalance.
- **Probing and interpretability** — what does the representation encode, and does the explanation match the mechanism?
- **Efficiency** — distillation, quantisation, pruning, measured honestly on the same hardware.
- **Evaluation itself** — does the metric measure what everyone assumes? Build a small human evaluation.
- **Genuine theory** — a bound, a convergence proof, an equivalence. Small and complete beats grand and sketched.

# Six places where gaps are written down

1. **Limitations sections.** Mandatory at ACL venues — authors list, in their own words, what their method cannot do. A topic menu.
2. **Future work** in recent papers and surveys.
3. **Survey taxonomies** — look for the boxes with one or two entries; that is where the field is thin.
4. **Workshop calls for papers** — this year's open problems, in bullet form.
5. **Task overview papers** — the closing section says exactly what nobody solved.
6. **Plateaued leaderboards**, or a simple baseline suspiciously close to the top.

## Two more

**GitHub issues** on popular repositories: "does not work for language X", "cannot reproduce Table 2".

**Failed reproductions** — a reproducibility report that did not work out is a thesis-sized question.

# The one-page proposal

Write it before any code — one hour, saves months

| Field | What to write |
|---|---|
| Question | One sentence, answerable, with a verb |
| Why it matters | Two sentences, no hype |
| Data | Name, size, licence, where it lives |
| Metric | The official one, plus what else you will report |
| Baselines | The trivial one and the strong one |
| Method | What you will try, in three bullets |
| Compute | Laptop / lab GPU / Colab — and whether it suffices |
| Milestones | Four dates: baseline, first result, freeze, draft |
| Risks & fallback | What could fail, and the thesis you still have if it does |

Part 2

# Tools, literature, assistants

# Your toolkit, week one

## References and writing

**Zotero** + browser connector + **Better BibTeX**: pinned keys and auto-export into your thesis `refs.bib`. A one-line note on every save. Cite the *published* version; clean BibTeX from DBLP or the ACL Anthology. Zotero flags retracted items; **Unpaywall** finds the free copy of a paywalled paper.

**Overleaf** for co-editing and supervisor comments (its AI assistant is metered); `git` + `latexmk` locally, one sentence per line. Ask for your department's template early. **Prism closes 1 January 2027** — do not start a thesis there, and export anything you have now.

## Code, data, compute

One git repository for everything, committed daily.

Free GPUs: Colab and Kaggle Notebooks (longer sessions). Ask about lab machines in week one — accounts take time.

Track runs (W&B / MLflow) with config, metrics and commit hash.

Check model and dataset **licences**. If your data involves people, raise it with your supervisor *before* collecting — GDPR is not optional.

# Getting a reference into your `.bib`

## The easiest route — no setup

1. Google Scholar: *Settings* → *Bibliography manager* → tick *Show links to import citations into BibTeX*. Once, ever.
2. A **Cite** link appears under every result → **BibTeX** → copy.
3. Paste into `refs.bib`, or into Zotero with *File* → *Import from clipboard*.

Bulk: star papers into *My library*, select, *Export* → *BibTeX*.

## Check what it gives you

"arXiv preprint" for a paper that appeared at a conference, missing pages, inconsistent venue names. Examiners read the bibliography.

**Cleaner, one click each: ACL Anthology · DBLP · arXiv** "Export BibTeX" · Semantic Scholar, ACM, IEEE, Springer.

**Any DOI, from the terminal:**

```
curl -LH "Accept:
application/x-bibtex"
https://doi.org/<DOI>
```

Past ∼20 references: connector + Better BibTeX auto-export does it all.

Protect capitals (`{BERT}`), authors as `Last, First`; run `bibtex-tidy` and check every DOI resolves before submitting.

# Finding and judging related work

| | |
|---|---|
| `"exact phrase"` | quote technical terms |
| `a OR b` | must be capitalised |
| `-survey` | exclude a term |
| `source:"SemEval"` | search inside a venue or task |
| `intitle:x` | precision filter |
| **Cited by → search within citing articles** | who applied it to *your* setting |

Also: ACL Anthology, Semantic Scholar, Connected Papers (week one), OpenReview — where you can read the actual reviews.

## Read in three passes

5 min (skim, decide) → 1 h (figures, method) → hours (re-derive). Pass 3 for the five papers your thesis depends on.

## Is this repository worth it?

Last commit, contributors, who answers issues. **Search the issues for "reproduce"** before trusting its numbers. Checkpoints, pinned deps, a LICENSE. Stars measure marketing, not correctness.

# Two Google tools: ask a question, get a first review

## Scholar Labs

scholar.google.com/scholar_labs

- A search mode inside Scholar, open to all signed-in users: type the *research question*, not keywords.
- Returns papers that answer it, with a sentence on how each does, and follow-up questions to narrow in.
- Good for the first week, when you do not know the vocabulary. Then read the papers — the explanation is a lead, not a citation.

## Deep Review

research.google/deepreview

- Google's reviewer-in-a-box: upload a PDF, it checks each section for maths errors, logical gaps and unsupported claims.
- Two free reviews a week; uploads are not used for training.
- Run it on your complete draft before your supervisor reads it. Read every flag: fix what is wrong, note why you reject the rest.
- **Your own draft only**, never someone else's unpublished work.

# Using an AI assistant honestly

## Use it for

- Explaining unfamiliar notation; walking a derivation.
- Getting someone else's repository to run.
- Boilerplate: loaders, plots, evaluation scripts, Colab setup.
- Refactoring before the code becomes unmaintainable.
- LaTeX debugging; tightening prose *you* wrote.
- Acting as a hostile examiner on your draft.

## Five rules

1. **You must be able to explain every line you submit.** The defence tests exactly this.
2. **Never cite a paper you have not opened.** Models invent references that look right. Fabricated citations fail theses.
3. **Never let it write your results or your analysis.**
4. **Follow your university's policy** and declare use if required — ask, do not guess.
5. **No confidential or personal data** in a cloud tool.

Part 3

# Doing the work

# The order that works

1. **Get the data and look at it by hand.** Fifty examples. Count the labels, find the duplicates.
2. **Build the dumbest baseline** and measure it — majority class, TF-IDF + logistic regression, a rule.
3. **Reproduce the strong baseline.**
4. **Only then** try your idea — one change at a time, each measured.
5. **Ablate.** Start early; ablations always take longer than the main result.
6. **Analyse the errors** by hand, and categorise them.

## A results table from day one

Date, commit, config, seed, dev score, and one line on what changed and why. Your results chapter becomes a formatting exercise rather than an archaeology project.

## Negative results are results

"I tried X, it did not help, here is the evidence and here is why" is a legitimate finding — provided the experiment was fair. Keep the failures in your log.

## Mistakes that cost marks

- **Test-set leakage** — tuning on test, duplicates shared between train and test, preprocessing fitted before the split.
- **One seed.** A single run with $+0.4$ is noise. Report mean $\pm$ standard deviation over at least three seeds.
- **Unfair comparison** — your method tuned for a week, the baseline run with defaults for an hour. Examiners spot this immediately.
- **Accuracy on imbalanced data.** Report macro-F1, per-class results, or a confusion matrix.
- **Results that cannot be regenerated** because the configuration lived in your shell history.
- **Plots without axis labels, units, or a caption** that says what the reader should conclude.

Part 4

# Writing and defending

# A structure that works

Start writing in month one, not month six

| Chapter | What goes in it | Pages |
|---|---|---|
| Abstract | Problem, what you did, headline result | 1 |
| Introduction | Context, the question, contributions as a list | 3–5 |
| Background | Only the concepts your chapters need | 5–10 |
| Related work | By *theme*, not one paragraph per paper | 5–8 |
| Method / system | Precise enough to reimplement | 5–10 |
| Experiments | Data, splits, metrics, baselines, results | 8–15 |
| Analysis | Ablations, error analysis, surprises | 4–8 |
| Conclusion | Findings, limitations, future work | 2–3 |

*Your department's required length and format always win.*

# Writing advice that actually helps

- **Write the figures and tables first**, then the text that explains them. If a table needs three paragraphs of excuses, the experiment is wrong, not the prose.
- **Every chapter opens** with one sentence saying what it does and closes with one saying what it showed.
- **Say what you did**, past tense, active voice, short sentences.
- **Define every symbol** the first time, and keep notation consistent.
- **Captions are self-contained.** A reader who looks only at your figures should still follow the thesis.
- **Cite properly, quote almost never.** Copying sentences — from papers, websites or a model — without quotation marks and a citation is plagiarism, and it is checked.
- **Right citation command:** `\citet` in the sentence, `\citep` as an aside (biblatex: `\textcite` / `\parencite`).
- **Give your supervisor a complete draft** at least three weeks before the deadline. Feedback on an incomplete draft is feedback on the wrong problem.

# The defence

- **Structure:** problem (2 slides) → why it is hard (1) → what you did (3–5) → results (3–4) → analysis and limitations (2) → conclusion (1). One idea per slide.
- **Rehearse out loud, timed, twice** — once alone, once in front of a person.
- **A 60-second demo**, recorded as a fallback. Never rely on live internet.
- **Know your numbers:** dataset sizes, your best score, the baseline, training time.

## Expect these questions

Why this baseline? What happens if the data changes? Why does it fail on class X? What would you do with six more months? What did you find hardest? Is the improvement significant?

"I do not know" is an acceptable answer when followed by "but here is how I would find out". Guessing is not.

Part 5

# Timetable and afterwards

# Six months

| Month | Done by the end of it |
|---|---|
| **1** | Topic agreed and proposal signed off; tools installed; data downloaded and inspected; 10–15 papers read; trivial baseline running |
| **2** | Strong baseline reproduced; experiment logging in place; background and related work drafted; task dates in the calendar |
| **3** | Your method implemented; first real results; method chapter drafted |
| **4** | Ablations and seeds; results tables generated by script; experiments chapter drafted |
| **5** | Error analysis; analysis chapter; **complete draft** to your supervisor; results frozen |
| **6** | Revisions; references checked one by one; defence slides; two rehearsals; submission |

## Weekly rhythm

Send five bullets the evening before every meeting: **what I did** · **what I found** · **what is blocked** · **what I propose** · **what I need from you**. "I am stuck" sent on Tuesday is worth three weeks of silence.

# After the thesis

- **Publish it if it is good** — a shared-task system paper, a workshop paper, or a student research workshop submission. A paper turns coursework into a credential.
- **Clean the repository** and make it public: a README that reproduces your main table, the environment file, a licence. The best item on an entry-level CV.
- **Put up a page** — `username.github.io` with AcademicPages takes an afternoon.
- **Ask for a reference letter while it is fresh**, and give your supervisor your abstract, results and plans to write from.
- **Considering a PhD?** The companion guide — *Starting a PhD in Machine Learning or NLP* — covers venues, deadlines and the research habits that follow from here.

# Checklists

## Week 1

Supervisor slot fixed · proposal agreed · Zotero + Better BibTeX · git repo with `notes/` · LaTeX project started · data downloaded, licence checked, 50 examples read · compute sorted · deadlines in the calendar

## Before freezing results

Trivial + strong baseline + your method on the same splits · 3+ seeds, mean ± std · test set used once · every table and figure regenerated by script · ablations done · errors categorised · code runs from a clean clone

## Before submitting

Every citation checked and every DOI resolves · `bibtex-tidy` run · department format met · figures legible in greyscale · limitations written honestly · a Deep Review pass on the final PDF · read-aloud pass · complete draft seen 3 weeks earlier · slides rehearsed twice with a recorded demo

## The shortest version

Pick a problem that already has data and a metric — a shared task is the easiest way to get one. Run the dumb baseline first. Change one thing at a time and write down what happened. Understand everything you submit. Write from month one, show your supervisor a complete draft early, and be honest about what did not work.

That is a good thesis.

Konstantinos Skianis ▪ University of Ioannina ▪ y3nk0.github.io