Doing an Undergraduate Thesis in Machine Learning or NLP

A Practical Guide

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

https://y3nk0.github.io

Written for final-year students starting a thesis in machine learning or natural language processing. It is deliberately concrete: how to find a topic that is neither trivial nor impossible, how to run experiments you can defend, how to write a thesis that reads well, and what to do with it afterwards. Nobody expects you to beat the state of the art. They expect you to do something correctly, understand it completely, and explain it clearly.

October 2026

Your department's own regulations — length, format, deadlines, grading — always take precedence over anything here.

Contents

1 What a thesis is, and what it is not **2**
1.1 Scope: the single most common failure . . . 2

2 Finding a topic: three routes **2**
2.1 Route A — shared tasks, challenges and SemEval . . . 3
2.2 Route B — an application thesis . . . 4
2.3 Route C — an analysis or theoretical thesis . . . 5
2.4 How to find a gap without reading a thousand papers . . . 5
2.5 The one-page proposal . . . 5

3 Your toolkit, set up in week one **6**
3.1 References: Zotero . . . 6
3.2 Writing: LaTeX and Overleaf . . . 7
3.3 Code, data and compute . . . 7

4 Finding and judging related work **7**
4.1 Google Scholar, properly . . . 8
4.2 Reading efficiently: three passes . . . 8
4.3 Is this work worth building on? . . . 8

5 Using an AI assistant honestly **9**
5.1 Set it up . . . 9
5.2 Use it for . . . 9
5.3 The rules . . . 9

6 Doing the work **9**
6.1 The order that works . . . 9
6.2 Mistakes that cost marks . . . 10
6.3 Keep a results table from day one . . . 10

7 Writing the thesis **10**
7.1 A structure that works . . . 10
7.2 Writing advice that actually helps . . . 11

8 The defence **11**

9 A timetable **11**

10 After the thesis **12**

11 Checklists **12**

A Link directory **13**

B One-page proposal template **14**

1 What a thesis is, and what it is not

An undergraduate thesis is a supervised piece of work, typically 4–9 months alongside your other courses, that shows you can take a research-level problem, work on it methodically, and communicate the result. It is the first time most students own a project end to end.

What examiners actually look for

- A **clear question**, stated in one sentence, that your work answers.
- **Correct methodology** — sane baselines, honest evaluation, no leakage, results you can reproduce.
- **Understanding.** You must be able to explain every design decision and every line you submit. This is what the defence tests.
- **Clear writing** — a reader from your department should understand what you did and why.
- **Honest limitations.** Saying what did not work, and why, is a sign of maturity, not of failure.

What is *not* required

Beating the state of the art. Inventing a new architecture. Training a large model from scratch. A thesis that carefully reproduces, analyses and extends existing work — and explains what it found — is a good thesis. A thesis that chases a record and ends with one unexplained number is not.

1.1 Scope: the single most common failure

Almost every thesis that goes badly goes badly for the same reason: the topic was too large, so the student spent six months getting a pipeline to run and had nothing to say about it. Use this test:

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

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

2 Finding a topic: three routes

You do not need an original idea to start. You need a *well-defined problem* with data and a way to measure success. There are three reliable routes to one; pick whichever fits how you like to work — but check what is already on offer before you start inventing.

Step zero: check what is already on offer

Most supervisors keep a list of open topics — on their web page, on the department's thesis board, or in a shared document — and the quickest good thesis usually starts there. The order that works:

1. **Look for advertised topics first.** Read the supervisor's page and the department's list, and ask the secretariat or older students where topics are posted.
2. **Pick one and do a small piece of research before the first meeting:** read the two or three papers the topic points to, find the dataset and the metric, skim or run one existing implementation, and write down three questions. Arriving with that turns the first meeting from an introduction into a plan.
3. **Only if you are given freedom**, or none of the advertised topics fits, use the three routes below to find your own.

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 theoretical thesis	Reproduce, probe, compare, measure: explain *why* methods behave as they do	You like understanding over building

2.1 Route A — shared tasks, challenges and SemEval

This is the strongest default for an undergraduate thesis, and it is under-used. A **shared task** is a community competition: organisers publish a precisely defined problem, training and development data, an official metric, and an evaluation window. Everyone is evaluated on the same hidden test set, and participants write a short **system description paper**.

Why this is such a good fit

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

Where to find them

SemEval (https://semeval.github.io) — the International Workshop on Semantic Evaluation, run annually with a dozen or more tasks: semantic similarity, sentiment and emotion, irony, humour, hate speech, factuality and hallucination detection, clinical and legal NLP, multilingual tasks, and more. Tasks are announced well ahead of the evaluation window and the workshop is co-located with a *ACL conference.

Workshop shared tasks at *ACL venues — most workshops at ACL, EMNLP, NAACL, COLING and EACL run their own task. Check https://aclanthology.org and the workshop lists on each conference site. Examples of recurring families: **WMT** (machine translation and evaluation metrics), **CoNLL** shared tasks, **BabyLM** (sample-efficient language modelling), **BioNLP**, **WNUT** (noisy user-generated text), **CheckThat!** and other fact-checking labs.

CLEF labs and TREC tracks — https://www.clef-initiative.eu, https://trec.nist.gov: information retrieval, biomedical question answering (**BioASQ**), argument mining, authorship, misinformation. Longer cycles, very well organised.

Competition tracks at ML conferences — the NeurIPS competition track and similar programmes at other venues: usually larger in scope, worth a look if you have compute.

Platforms — **Kaggle** (https://www.kaggle.com), **CodaBench / CodaLab**, **EvalAI**, **AIcrowd**, **DrivenData**. Kaggle is excellent for learning, but a Kaggle competition alone is usually not a thesis: you still need a research question and an analysis chapter around it.

Leaderboards and benchmark hubs — Hugging Face (https://huggingface.co) hosts datasets, models and evaluation leaderboards; many tasks keep a permanent leaderboard open long after the official window closes.

> **Check the dates before you commit**
>
> Shared tasks run on fixed cycles — registration, training data release, development phase, a short evaluation window on the hidden test set, then the system paper deadline — and the exact dates change every edition. Read the task's own page, write the four dates into your calendar, and confirm with your supervisor that they fit your thesis deadline. An evaluation window that opens after your submission date is a thesis you cannot finish.

How to run a shared task as a thesis

1. **Pick two or three candidate tasks** and read last year's *task overview paper* plus three or four system papers. Thirty minutes each. You will immediately see what a good system looks like and where the easy gains are.
2. **Reproduce the organisers' baseline first.** Before anything clever. If you cannot match the baseline, nothing else you do is interpretable.
3. **Build your own validation split** and trust it. The leaderboard has limited submissions; your development set does not.
4. **Keep an experiment log from run one** — configuration, seed, score, what you changed, what you expected. This log becomes your results chapter.
5. **Submit something early**, even a weak system. Submission pipelines always break the first time; find out in week three, not on the last day.
6. **Write the system paper** with your supervisor. Then the thesis is the long version: the paper plus the background, the failed attempts, the ablations and the analysis that did not fit in four pages.

> **If the task window has closed**
>
> Use it anyway. Past-edition data, metrics and leaderboards are usually public, so you can run the whole thing retrospectively and compare against the published table of participants. You lose the live leaderboard; you keep every other advantage — and you can report where your system *would* have ranked.

2.2 Route B — an application thesis

Take a method that is known to work and apply it, carefully, somewhere it has not been applied. The contribution is the careful application, the evaluation and the analysis — not the method.

Good sources of a target:

- **A language or dialect with few resources.** Most NLP results are reported on English. Greek, and most other languages, have far fewer evaluated baselines — building and *properly evaluating* one for a specific task is a real contribution.
- **A domain with its own vocabulary:** legal, medical, financial, educational, agricultural, archival text; code; sensor or log data.
- **A deployment constraint:** the model must run on a laptop, a phone or a Raspberry Pi; must answer in under 100 ms; must work with 50 labelled examples; must run without sending data to a cloud API.
- **A real user with a real problem** — a lab in another department, a university service, a local company, a public dataset nobody has modelled. Scope it tightly and get the data question answered in week one.

> **What makes an application thesis rigorous rather than a demo**
>
> - At least two baselines, one of them embarrassingly simple (majority class, TF-IDF + logistic regression, nearest neighbour, a rule). If the simple baseline wins, that *is* your finding — report it.

- A fixed split made *before* you look at results, and a test set touched once.
- Several seeds, with mean and spread. One run proves nothing.
- An error analysis: categorise 50–100 mistakes by hand. This is the single most valuable chapter in most application theses, and the one examiners remember.
- A cost column: parameters, training time, inference latency, hardware. Accuracy per euro and per second is a real axis.

2.3 Route C — an analysis or theoretical thesis

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.** Take a well-known recent result and check it: other seeds, other datasets, other languages, other hyperparameters. Reproducibility studies are a recognised genre with their own venues and tracks.
- **Ablation and attribution.** A paper proposes five components and reports one number. Which component actually matters? Very often the answer is "not the one in the title".
- **Robustness and failure modes.** What happens under typos, dialect, code-switching, domain shift, adversarial rewrites, longer inputs, class imbalance?
- **Probing and interpretability.** What does the representation encode? Where does the behaviour live in the network? Does the explanation match the mechanism?
- **Efficiency.** Distillation, quantisation, pruning, retrieval instead of parameters — measured honestly on the same hardware.
- **Evaluation itself.** Does the metric measure what everyone assumes? Build a small, careful human evaluation and compare. Metric critiques are consistently under-supplied and over-needed.
- **Genuine theory** — a proof of convergence, a bound, a complexity result, an equivalence between two formulations — if you enjoy mathematics and have a supervisor to work with closely. Keep the claim small and complete rather than grand and sketched.

2.4 How to find a gap without reading a thousand papers

Six places where gaps are written down explicitly

1. **"Limitations" sections.** Every ACL-venue paper must have one. Authors list, in their own words, what their method cannot do. That list is a topic menu.
2. **"Future work"** in the conclusion of recent papers and surveys.
3. **Survey taxonomies.** Find the most recent survey in the area and look for the boxes with one or two entries — that is where the field is thin.
4. **Workshop calls for papers.** They list the open problems the community cares about this year, in bullet form.
5. **Task overview papers.** The closing section of a shared-task overview says exactly what nobody solved.
6. **Leaderboards that have plateaued**, or where a simple baseline is suspiciously close to the top — a sign the task, the metric or the data deserves a second look.

Two more practical sources: **GitHub issues** on popular repositories ("does not work for language X", "cannot reproduce Table 2"), and **reproducibility reports** that failed — a failed reproduction is a thesis-sized question.

2.5 The one-page proposal

Before you write any code, write one page and send it to your supervisor. It takes an hour and 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, is it public
Metric	The official one, plus what you will also report
Baselines	The trivial one and the strong one
Method	What you will try, in three bullets
Compute	Your 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

3 Your toolkit, set up in week one

3.1 References: Zotero

You will read 40–100 papers. Without a system you will re-find the same paper five times and mis-cite one in your thesis.

- **Zotero** (https://www.zotero.org), free, plus the **browser connector**: one click on an arXiv, ACL Anthology or Scholar page saves the metadata *and* the PDF.
- **Better BibTeX** (https://retorque.re/zotero-better-bibtex/): stable citation keys you *pin* (`devlin2019bert`), and **auto-export** of your library into the `refs.bib` of your thesis repository. Your bibliography is then always current and you never hand-edit BibTeX.
- Take a one-line note on every save: *what it does, why I care*. In month six that is the difference between a library and a junkyard.
- Cite the **published version**, not the preprint, when one exists. Get clean BibTeX from the **ACL Anthology** (https://aclanthology.org) or **DBLP** (https://dblp.org) — Google Scholar's is often sloppy about venue names and pages.
- Zotero also checks your library against Retraction Watch: a red banner on an item means it was retracted. Find out why before citing.
- **Unpaywall** (https://unpaywall.org/products/extension), a browser extension that finds the legal free copy of a paywalled paper — essential if your university's access is patchy.

Getting a reference into your `.bib`

The easiest route needs no setup at all:

1. In Google Scholar, open *Settings* → *Bibliography manager* and tick *Show links to import citations into BibTeX*. One time only.
2. A **Cite** link now appears under every result. Click it, then **BibTeX**, and copy the entry.
3. Paste it into your `refs.bib` — or into Zotero with *File* → *Import from clipboard*, which keeps your library and your bibliography in step. Scholar can also export in bulk: star papers into *My library*, select them, then *Export* → *BibTeX*.

> **Check whatever Scholar gives you**
>
> Scholar's entries are often sloppy: `arXiv preprint arXiv:1810.04805` for a paper published at NAACL, missing pages and editors, venue names that differ between entries, mangled accents. A bibliography full of "arXiv preprint" is the clearest sign that nobody checked. Cleaner sources, each one click away:
>
> - **ACL Anthology** — a BibTeX box on every paper page, and a full `.bib` for each volume.
> - **DBLP** — per-paper BibTeX and a whole-author export.
> - **arXiv** — "Export BibTeX citation" on the abstract page.
> - **Semantic Scholar**, ACM, IEEE and Springer pages all have a cite or export button.

- **Any DOI, from the terminal:** `curl -LH "Accept: application/x-bibtex" https://doi.org/<DOI>` returns the publisher's own entry.

Past twenty or so references, stop copy-pasting: the Zotero connector plus Better BibTeX auto-export does all of this for you and keeps your keys stable.

Two things every tool gets wrong unless you tell it: protect capitals in titles (`{BERT}`, `{Greek}`) or the bibliography style will lowercase them, and write authors as `Last, First and Last, First`. Before you submit, run `bibtex-tidy` (https://github.com/FlamingTempura/bibtex-tidy) over `refs.bib` to format, sort and deduplicate it, and check that every DOI resolves — examiners now look for invented references.

3.2 Writing: LaTeX and Overleaf

Write the thesis in LaTeX unless your department requires otherwise. Learn it in a weekend and it never costs you again. Typst is nice, but your department's template and every venue's are LaTeX.

- **Overleaf** (https://www.overleaf.com) — browser-based, nothing to install, real-time co-editing and comments. Your supervisor can read and annotate the draft without exchanging files. Check whether your university has an institutional licence before paying. It now has a built-in, metered AI assistant as well; the rules in §5 apply to it exactly.
- **Prism** (https://prism.openai.com) — OpenAI's AI-native LaTeX workspace **shuts down on 1 January 2027**. Do not start your thesis there. If you already have projects, export them now: exports do not include comments, version history, collaborators or linked integrations.
- **Locally:** `git` + `latexmk -pdf -pvc` + VS Code. Write **one sentence per line** — it makes diffs readable and revisions painless; `latexdiff` shows your supervisor exactly what changed between drafts.
- Start from your department's thesis template if one exists; otherwise a clean `report` or `book` class is fine. Ask early about required title-page wording, logos and declarations.

3.3 Code, data and compute

- **One git repository** for the whole thesis: code, configs, notes, and the LaTeX. Commit daily. "It was working yesterday" is only recoverable if yesterday is in git.
- **Free GPUs:** Google Colab and Kaggle Notebooks both give usable GPU time; Kaggle sessions are longer and more predictable. Ask your supervisor about lab machines or university clusters — and ask in week one, because accounts take time.
- **Hugging Face** (https://huggingface.co) for pretrained models and datasets. Check the **licence** of every model and dataset you use, and say in the thesis which you used and under what terms.
- **Experiment tracking:** Weights & Biases, MLflow or TensorBoard. Log the config, the metrics and the git commit for every run. "Which run produced Table 4?" must have an answer in month six.
- **Data protection:** if your data involves people — scraped social media, user records, recordings, surveys — raise it with your supervisor before you collect anything. Licensing and GDPR are not optional, and "I found it online" is not a licence.

4 Finding and judging related work

4.1 Google Scholar, properly

Query / feature	What it does
`"hate speech detection"`	Exact phrase. Always quote technical terms.
`greek OR hellenic`	`OR` must be capitalised.
`-survey`	Excludes a term.
`author:"K Skianis"`	Author search.
`source:"SemEval"`	Restricts to a publication whose name contains the phrase — the way to search inside a venue or a task.
`intitle:transformer`	Term must appear in the title: a brutal, effective precision filter.
Custom range + Sort by date	This month's work rather than 2019's most-cited.
Cited by → search within citing articles	Search *inside* the papers citing a key work. The fastest way to find who applied a method to *your* setting.
All N versions	Finds the free PDF and the published version.

Set up two or three **alerts** — one for your topic, one for citations of the paper your work builds on — and triage them into Zotero once a week.

Beyond Scholar: the **ACL Anthology** (every *ACL paper, free, full-text search), **Semantic Scholar** (TLDR summaries, citation intent), **Connected Papers** (https://www.connectedpapers.com, a visual neighbourhood from one seed paper — excellent in week one), **OpenReview** (https://openreview.net, where you can read the actual *reviews* of papers), and **paperswithcode.co**.

Scholar Labs. https://scholar.google.com/scholar_labs/search is a search mode inside Scholar, open to all signed-in users: type the research question itself rather than keywords ("does data augmentation help low-resource Greek NER?") and it returns papers that answer it, with a sentence on how each one does and follow-up questions to narrow in. Good for the first week, when you do not know the vocabulary yet; then read the papers — the explanation is a lead, not a citation.

4.2 Reading efficiently: three passes

> **Keshav's method, which will save you weeks**
>
> **Pass 1** (5 minutes): title, abstract, introduction, section headings, conclusions — then decide whether to continue. **Pass 2** (1 hour): figures, tables and the method, to the level where you could summarise it. **Pass 3** (several hours, rare): re-derive the work. Give pass 3 to the five papers your thesis depends on, pass 1 to a hundred.

4.3 Is this work worth building on?

- **Citations, normalised by age.** A 2026 paper with 40 citations may be hotter than a 2019 paper with 400. Scholar's per-year bar chart shows whether interest is accelerating or dead.
- **Who cites it** matters more than how many. Ten citations from strong groups beat 200 from unrelated surveys.
- **GitHub signals:** last commit date, number of contributors, who answers issues, whether releases and pinned dependencies exist, and whether checkpoints are provided. *Search the issue tracker for "reproduce"* before you trust a repository's numbers — do this every time.
- **Stars measure marketing, not correctness.** The only real test is running it yourself on a task you understand. Use https://star-history.com to tell a one-day launch spike from steady adoption.
- **Licence.** Research-only or non-commercial terms can restrict what you are allowed to do and publish. Check before you invest a month.

5 Using an AI assistant honestly

An AI assistant will speed up the mechanical parts of your thesis considerably. It can also end your degree if you use it to replace the thinking the thesis is meant to demonstrate. Both of those are true at the same time.

5.1 Set it up

- A coding assistant in your thesis repository, with a short project memory file stating your conventions: Python version, how to launch training, where data lives, plotting style.
- Connect it to your filesystem and, if you can, to your Zotero library and literature APIs — an assistant that reads your actual PDFs is a different tool from one that guesses.
- Keep a dated `notes/` folder it can read and write: decisions, failed runs, hyperparameters.

5.2 Use it for

- Explaining unfamiliar notation and walking through a derivation.
- Getting someone else's repository to run.
- Boilerplate: data loaders, plotting, evaluation scripts, Slurm or Colab setup.
- Refactoring code before it becomes unmaintainable.
- LaTeX debugging and table formatting.
- Tightening prose you have already written.
- Acting as a hostile examiner on your draft — and, for a free first review of the PDF, Google's Deep Review (box below).

A free first review of your draft: Deep Review

Google's **Deep Review** (https://research.google/deepreview/) reads a PDF like a careful examiner: it checks each section for mathematical errors, logical gaps and unsupported claims, then the whole for consistency. Two reviews a week are free, and Google states that uploads are not used to train its models. Run it on your complete draft before your supervisor reads it, then read every flag: fix what is wrong and note why you reject the rest. It is one more hostile reader, not a verdict — and never upload someone else's unpublished work.

5.3 The rules

Five rules, and the first one is not negotiable

1. **You must be able to explain every line you submit.** At the defence you will be asked why a function is there and what a number means. "The assistant wrote it" ends the conversation badly. If you do not understand it, do not submit it.
2. **Never cite a paper you have not opened.** Models invent plausible references — right-looking authors, venue and year, all fabricated. Every entry in your bibliography must come from Zotero, DBLP or the Anthology after you have read at least the abstract. Fabricated references are the fastest way to fail a thesis.
3. **Never let it write your results or your analysis.** Numbers come from runs you did; interpretation comes from you.
4. **Follow your university's policy** on AI assistance and declare your use if a declaration is required. Ask your supervisor what the rule is in your department — do not guess.
5. **Do not paste confidential or personal data** — someone's unpublished manuscript, data under an NDA, or personal records — into a cloud tool.

6 Doing the work

6.1 The order that works

1. **Get the data and look at it by hand.** Open fifty examples. Count the labels. Find the duplicates and the empty rows. One hour here saves weeks of confusion later.

2. **Build the dumbest possible baseline** and measure it. Majority class, TF-IDF + logistic regression, a rule. This number anchors everything that follows.
3. **Reproduce the strong baseline** from the paper or the task.
4. **Only then** try your idea — one change at a time, each measured.
5. **Ablate.** Remove each piece of your method and show what it was worth. Ablations always take longer than the main result; start them early.
6. **Analyse the errors** by hand, and categorise them.

6.2 Mistakes that cost marks

The usual suspects

- **Test-set leakage** — tuning on test, duplicates shared between train and test, preprocessing or feature selection fitted on the full dataset before splitting.
- **One seed.** A single run with a +0.4 improvement is noise. Report mean ± 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.

6.3 Keep a results table from day one

One spreadsheet or CSV: date, git commit, config file, seed, dev score, test score (only when you finally run it), and a one-line note on what changed and why. Your results chapter is then a formatting exercise rather than an archaeology project, and the final table regenerates from logged numbers with a script.

Negative results are results

"I tried X; it did not help; here is the evidence and here is why I think it failed" is a legitimate and valuable thesis finding — provided the experiment was fair and the analysis is honest. Many strong theses are built exactly this way. Keep the failed experiments in your log; do not delete them.

7 Writing the thesis

Start writing in month one, not month six. Writing is how you discover that your story has a hole, and holes are cheap to fix early.

7.1 A structure that works

Chapter	What goes in it	Rough size
Abstract	Problem, what you did, headline result, in one paragraph	1 page
Introduction	Context, the question, your contributions as a bullet list, thesis outline	3–5
Background	Only the concepts the reader needs for your chapters	5–10
Related work	Organised by *theme*, not one paragraph per paper; end by saying where your work sits	5–8
Method / system	What you built, precisely enough to reimplement	5–10
Experiments	Data, splits, metrics, baselines, setup, then results tables	8–15
Analysis	Ablations, error analysis, what surprised you	4–8
Conclusion	What you found, limitations, future work	2–3
References, appendix	Bibliography; extra tables, prompts, hyperparameters	—

Follow your department's required length and format; the sizes above are a sane default, not a rule.

7.2 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, not what one might do.** Past tense, active voice, short sentences. "We tokenise with X" beats "it was deemed appropriate to employ tokenisation".
- **Define every symbol** the first time it appears, and keep notation consistent across chapters.
- **Captions are self-contained.** A reader who looks only at your figures should still understand the thesis.
- **Cite properly and quote almost never.** Copying sentences — from papers, from websites, or from a model — without quotation marks and a citation is plagiarism, and it is checked. Paraphrase in your own words and cite the source.
- **Use the right citation command.** `\citet{}` when the authors are part of the sentence ("Devlin et al. (2019) show"), `\citep{}` when the citation is an aside ("(Devlin et al., 2019)"); with biblatex, `\textcite` and `\parencite`. Mixing them is the most common formatting slip in student theses.
- **Give your supervisor a complete draft**, not a chapter, at least three weeks before the deadline. Feedback on an incomplete draft is feedback on the wrong problem.

8 The defence

Most departments ask for a 15–25 minute presentation plus questions. It is not an interrogation; it is you explaining your work to people who want you to succeed.

- **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. Speaking it aloud exposes everything you do not actually understand.
- **A short demo** — 60 seconds, recorded as a fallback — is worth a hundred words if you built a system. Never rely on live internet.
- **Know your numbers:** dataset sizes, your best score, the baseline's score, and how long training took.
- **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.

9 A timetable

A six-month plan; stretch or compress it to your own deadline, and put every milestone in a calendar.

Month	What is done by the end of it
1	Topic agreed and one-page proposal signed off; tools installed (Zotero, git, Overleaf, assistant); data downloaded and inspected; 10–15 papers read; trivial baseline running.
2	Strong baseline reproduced; experiment logging in place; background and related-work chapters drafted; shared-task dates in the calendar (if Route A).
3	Your method implemented; first real results; method chapter drafted.
4	Ablations and seeds; results tables generated by script; first full draft of the experiments chapter.
5	Error analysis; analysis chapter; **complete draft** to your supervisor; results frozen.
6	Revisions, polish, references checked one by one, defence slides, two rehearsals, submission.

Weekly rhythm with your supervisor

Meet weekly or fortnightly, and send five bullets the evening before: **what I did** · **what I found** · **what is blocked** · **what I propose next** · **what I need from you**. Keep a shared running document. Supervisors help far more effectively when the question is specific, and "I am stuck" sent on Tuesday is worth three weeks of silence.

10 After the thesis

- **Publish it if it is good.** A shared-task system paper, a workshop paper, or a student research workshop submission (most *ACL conferences have one) is realistic for a strong thesis — ask your supervisor. A paper transforms a thesis from coursework into a credential.
- **Clean the repository** and make it public: a README that explains how to reproduce your main table, the environment file, and a licence. This is the single best item on an entry-level CV.
- **Put up a page** — `username.github.io` with the AcademicPages template takes an afternoon: a short bio, your thesis PDF, your code, your CV. Employers and MSc admissions committees search your name.
- **Ask for a reference letter while everything is fresh**, and give your supervisor your thesis abstract, your results and your plans to write from.
- **If you are 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.

11 Checklists

Week 1

- Supervisor meeting slot fixed
- One-page proposal written and agreed
- Zotero + connector + Better BibTeX installed
- Git repository created, with `notes/` and a README
- Overleaf (or local LaTeX) project started
- Data downloaded, licence checked, 50 examples read by hand
- Compute sorted: lab machine, cluster account, or Colab/Kaggle
- Deadlines in the calendar, including any task dates

Before you freeze results

- Trivial baseline, strong baseline and your method, all on the same splits
- At least three seeds, mean ± std reported
- Test set used once; no tuning on it
- Every table and figure regenerated by a script from logged results
- Ablations done; error analysis categorised
- Code runs from a clean clone, following your own README

Before you submit

- Every citation checked against the real paper and every DOI resolves — no invented references; `bibtex-tidy` run
- Department's format, length and declaration requirements met
- Figures legible in print and greyscale; captions self-contained
- Limitations section written honestly
- A Deep Review pass on the final PDF; every flag fixed or consciously rejected
- Spell-check and a full read-aloud pass
- A complete draft seen by your supervisor at least three weeks earlier
- Slides rehearsed twice, timed, with a recorded demo as a fallback

A Link directory

What	Where
Shared tasks and challenges	
SemEval	https://semeval.github.io
ACL Anthology (workshops)	https://aclanthology.org
CLEF initiative	https://www.clef-initiative.eu
TREC	https://trec.nist.gov
Kaggle	https://www.kaggle.com
CodaBench / CodaLab	https://www.codabench.org
EvalAI	https://eval.ai
AIcrowd	https://www.aicrowd.com
DrivenData	https://www.drivendata.org
Conference deadlines	https://paperswithcode.co/ai-deadlines
	http://aideadlines.org/?sub=ML,CV,NLP
Tools	
Zotero	https://www.zotero.org
Better BibTeX	https://retorque.re/zotero-better-bibtex/
bibtex-tidy	https://github.com/FlamingTempura/bibtex-tidy
Overleaf	https://www.overleaf.com
Prism (closes 1 Jan 2027)	https://prism.openai.com
Unpaywall	https://unpaywall.org/products/extension
Deep Review (Google)	https://research.google/deepreview/
Hugging Face	https://huggingface.co
Google Colab	https://colab.research.google.com
Kaggle Notebooks	https://www.kaggle.com/code
Weights & Biases	https://wandb.ai
Literature	
Google Scholar	https://scholar.google.com
Scholar Labs	https://scholar.google.com/scholar_labs/search

What	Where
Semantic Scholar	https://www.semanticscholar.org
DBLP	https://dblp.org
OpenReview	https://openreview.net
Connected Papers	https://www.connectedpapers.com
Papers with Code	https://paperswithcode.co
Afterwards	
AcademicPages template	https://github.com/academicpages/academicpages.github.io
The PhD companion guide	https://y3nk0.github.io/guides/

B One-page proposal template

Title. Working title, one line.

Question. The one sentence this thesis answers.

Motivation. Two sentences: who cares, and why it is not already solved.

Route. Shared task / application / analysis (§2).

Data. Name, size, source, licence, splits.

Metric. Official metric, plus anything else you will report.

Baselines. The trivial one; the strong one; where their numbers come from.

Method. Three bullets on what you will try.

Compute. Hardware, and the largest model it allows.

Milestones. Baseline by _____; first result by _____; freeze by _____; full draft by _____.

Risks. What could fail.

Fallback. The thesis you still have if it does.

> **The shortest version of all of this**
>
> 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.