Your Title (simple, specific, and informative is enough; do not be vague or general and do not try to sound catchy)

Team Member Name 1
netid1

Team Member Name 2
netid2

Abstract

Concisely summarize the sections below.[1]

1 Problem

Precisely define the problem and the scope to be considered in the project. You need serious brainstorming. Possible scenarios include: (1) you already have a legitimate research problem that you want to work on for the project, (2) you can consider combining one idea with another (that have never been combined before), (3) you can find little tweaks or other experiments not been done yet in an existing work, (4) you choose a predefined project (limited).

2 Goal

Define the specific goal/hypothesis of the project (the more details the better).[2] Good examples:

- Investigate a brand new exciting machine learning model or algorithm recently proposed at AISTATS, ICLR, ICML, NeurIPS, or UAI that has never been applied to NLP before to improve a specific and relevant NLP task over existing baselines. Reasons this is good: technically challenging and not been done before.
- Reimplement a technically challenging NLP paper from scratch and replicate its results (ideally with additional exploration). This is only an option if the paper does not have publicly available code (e.g., on GitHub); we will check and reject the project if we find out that there is an implementation already available. Reasons this is good: also technically challenging and provides a useful implementation for people.

Bad examples:

- Use BERT for analyzing financial news (too vague/generic, also not technically challenging)
- Improve machine translation (how?)

Some real-world examples (course projects at Stanford): https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports.html.

3 Achievability

Describe in detail how you will actually achieve what you propose. Important considerations:

- Computing resources: do you have the computing power (e.g., GPUs) to execute your proposed experiments? How long does it take to train the models?
- Availability of data: in general, do not try to collect your own data. Significant time investment (at your own risk).
- Code framework: Python > 3.6 with PyTorch > 1.0 should be enough.
- You are learning a model, that is a function. What is your function (i.e., architecture)? What is the input and output of the function? What are the parameters of the model?
- Supervision setting: are you using labeled or unlabeled data or both?

[1] The content is heavily adapted from the project guideline at Princeton: https://nlp.cs.princeton.edu/cos484/projects.html.

[2] Even if you are doing a predefined project you must formulate a goal in your own terms.

4 Related Work

Do a thorough literature search. It is almost certain that someone has attempted something identical or similar before you. Start by Googling several variants of keyword (e.g., "neural style transfer NLP" and also "adapting models for different writing styles NLP"). It is okay if there is a similar existing work, if what you propose has something to add to it: **but you must explain how your work is distinguished from that work in what ways**.

Doing a literature search is also a good way to find an inspiration. For example take a look at a pick of top 100 NLP papers: https://github.com/mhagiwara/100-nlp-papers. Also a list of state-of-the-art methods with code: https://paperswithcode.com/sota. These are all influential papers. You cannot just repeat them unless you can justify as discussed above, but you can get a sense of what good NLP papers are like and track their recent followup works (see recent works that cite them on Google Scholar).

5 Tips (no need to include this in submission)

Clearly divide work between team members for optimal progress. **Start early and work on it every 1-2 days rather than rush at the end.** Set up work flow asap: download data, verify data, set up base code. Have running code and fully trained baseline model by milestone. Have a clear, well-defined hypothesis to be tested. Have meaningful tables, plots to display the key results.

Types of projects. Not exhaustive

- Experiment with improving an architecture on a predefined task
- Case study: Apply an architecture to a dataset in the real world (that has not been done before)
- Compete in a predefined competition (SemEval 2020, Kaggle, etc.)
- Stress test or comparison study of known models/architecture (e.g. when are RNNs better than Transformers for task XYZ?)
- Design a novel NN layer, objective function, optimizer, etc.
- Multi-modal tasks (RL + NLP, CV + NLP, etc.)
- Visualization/interpretability study of deep learning models

How to read papers. Do not read from start to finish in order. Spend the majority of time focusing on technical parts: deriving equations, verifying theorems, reading numbers in a table and understanding empirical behaviors. Avoid spending too much time reading text. Focusing on the abstract, the technical section, and the experiments section is usually enough; after that you can go to the introduction and other sections. Of course if you are not familiar with the problem at all and need to understand motivations, you may need to read the paper in order. You probably need to take multiple careful readings to understand a dense paper. If the paper has code available, checking the code is a good way to verify details.

Scenarios to avoid.

- Data not available or hard to get access to, which stalls progress
- All experiments run with prepackaged source, no extra code written for model/data processing
- Team starts late, only draft of code up by milestone
- Just ran model once or twice on the data and reported results (not much hyperparameter search done)
- A few standard graphs: loss curves, accuracy, without any analysis
- Results/Conclusion dont say much besides that it didnt work (even if results are negative, analyze them)

Milestone goals.

- Have code up and running
- Source of data explained correctly, along with true train/test/val split
- Acknowledge what Github repo, or other code you are basing off of
- Run baseline model and have results
- Brief discussion of initial, preliminary results
- Reasonable literature review (2+ related papers)
- 1-2 page progress report (not very formal)

What makes a good paper. A good paper is a paper from which the reader "learns a lot". While simple, this is packed with significance. The reader learns a lot because the paper proposes a refreshing, creative, shockingly unexpected idea; because the paper develops a technically daunting framework (something not everyone is qualified to achieve) and explains it in simple, approachable, yet rigorous terms; because the paper executes an idea with excellence and reveals nontrivial empirical findings; because the paper analyzes its results postmortem thoroughly and insightfully.

A good paper becomes great if it achieves a combination of these.