**CS 7375/W01 Spring 2022**

**Project Guidelines**

1/27/2022

In the project, you will work in groups of **one, two, three, or four** to apply the techniques that you've learned in this class to a new setting that you're interested in. Note that the larger the group, the higher the expectations for the project.

You will build a system to solve a well-defined task. Which task you choose is completely open-ended, but the methods you use should draw on the ones from the course.

You are encouraged to come to office hours often to discuss your project. Note that it will take several iterations to find the right project, so be patient; this exploration is an essential part of research, so learn from it. Have fun and don't wait until the last minute!

The final project should consist of the following stages:

* **Task definition**: What does your system do (what is its input and output)? What real-world problem does this system try to solve?

Make sure that the **scope** of the project is not too narrow or broad. For example, building a system to answer any natural language question is too broad, whereas answering short factoid questions about movies is more reasonable. This is probably the most important part of the project, but is also the part that you will not get practice doing from homeworks.

The first task you might come up with is to apply binary classification on some standard dataset. This is probably not enough. If you are thinking in terms of binary classification, you are probably thinking too narrowly about the task. For example, when recommending news articles, you might not want to make predictions for individual articles, but might benefit from choosing a diverse set of articles.

An important part of defining the task is the **evaluation**. In other words, how will you measure the success of your system? For this, you need to obtain a reasonably sized **dataset** of example input-output pairs, either from existing sources, or collecting one from scratch. A natural evaluation metric is accuracy, but it could be memory or running time. How big the dataset depends on your task at hand.

* **Approach**: Identify the challenges of building the system and the phenomena in the data that you're trying to capture. How should you model the task (e.g., using search, machine learning, logic, etc.)? There will be many ways to do this, but you should pick one or two and explain how the methods address the challenges as well as any pros and cons. What algorithms are appropriate for handling the models that you came up with, and what are the tradeoffs between accuracy and efficiency? Are there any implementation choices specific to your problem?

Before developing your primary approach, you should implement **baselines**. It is important as it gives you intuition for how easy or hard the problem you're solving is. Intuitively, baselines give lower bounds on the performance you will obtain. Importantly, baselines should be relatively easy to implement and can be done before you invest a lot of time in implementing a fancier approach. This can prune out problems early and save you a lot of time!

Baselines are simple algorithms, which might include using a small set of hand-crafted rules, training a simple classifier, etc. (Note that baselines are extremely simple, but you might be surprised at how effective they are.) While a predictor that guesses randomly provides a lower bound (and can be reported in the paper), it is too simple and doesn't give you much information. Predicting the majority label is a slightly less trivial baseline, and whether it's acceptable depends on how insightful it is. For classification, if the different labels have very different proportions, then it could be useful; otherwise it won't be. You are encouraged to have multiple baselines.

Overall, the main point of baselines is to get you to look at the problem carefully and think about what's possible. The accuracies of state-of-the-art systems on the dataset could be a baseline.

* **Literature review**: Have there been other attempts to build such a system? Compare and contrast your approach with existing work, citing the relevant papers. The comparison should be more than just high-level descriptions. You should try to fit your work and other work into the same framework. Are the two approaches complementary, orthogonal, or contradictory?
* **Error analysis**: Design a few experiments to show the properties (both pros and cons) of your system. For example, if your system is supposed to deal with graphs with lots of cycles, then construct both examples with lots of cycles and ones without to test your hypothesis. Each experiment should ask a concise question, such as: *Do we need to model the interactions between the ghosts in Pac-Man?* or *How well does the system scale up to large datasets?* Analyze the data and show either graphs or tables to illustrate your point. What's the take-away message? Were there any surprises?

**Milestones**

Throughout the quarter, there will be several milestones so that you can get adequate feedback on the project.

* **Build Your Team:** Each team can have up to 4 students. Please go to D2L --> click “Others” --> Groups. Try to enroll in ONLY one team you want. If you prefer to prefer to build a team with less than 4 students, please send me an email and tell me the member information (xzhang48@kennesaw.edu).

**DUE DATE: Friday 2/4 11:59pm. (Week 4)**

* **Proposal** (10% Points) (2 pages max): Define the input-output behavior of the system and the scope of the project. What is your evaluation metric for success? Collect some preliminary data, and give **concrete examples of inputs and outputs**. Implement a baseline. What are the challenges? Which topics (e.g., search, MDPs, etc.) might be able to address those challenges (at a high-level, since we haven't covered any techniques in detail at this point)? Search the Internet for similar projects and mention the related work.

**DUE DATE: Friday 2/25 11:59pm. (Week 7)**

* **Progress report** (20% Points) (4 pages): Propose a model and an algorithm for tackling your task. You should describe the model and algorithm in detail and use a concrete example to demonstrate how the model and algorithm work. Don't describe methods in general; describe precisely how they apply to your problem (what are the variables, factors, states, etc.)? You should also have finished implementing a preliminary version of your algorithm (maybe it's not fully optimized yet and it doesn't have all the features you want). Report your initial experimental results.

**DUE DATE: Friday 3/25 11:59pm. (Week 11)**

* **Presentation session** **[video presentation]** (20% Points): By the presentation session, you should have finished implementation, run a good chunk of your experiments, and done some basic error analysis. In the presentation, you should describe the motivation, problem definition, challenges, approaches, results, and analysis. The goal of the presentation is to convey the important high-level ideas and give intuition rather than be a super-detailed specification of everything you did (but you should still be precise). You will be evaluated on the contents of the presentation. The video should be no more than 15 minutes.

**DUE DATE: Friday 4/22 11:59pm. (Week 15)**

* **Presentation session peer review**: In the last lecture of this semester, I will post all the presentations of each group online. Based on your wanderings, you should choose up to 3 presentations that you liked the most and write a sentence or two describing each presentation and why you liked it. These reviews will not influence anyone's grade, but is just a fun way to encourage you to engage in the presentation session.

**NO DUE DATE: Simply add an extra part in your report to talk about the presentations you are interested.**

* **Final report** (50% Points) (5-10 pages): You should have completed everything (task definition, approach, literature review, error analysis).

**DUE DATE: Sunday 5/1 11:59pm. (Week 16)**

Note: you can have an appendix beyond the maximum number of allowed pages with any figures/plots/examples if you need.

**Submission**

Submit all milestones using D2L, I will set up a group assignment for each milestone. Please use the following name for each milestone.

* TeamNumber\_proposal.pdf, TeamNumber\_progress.pdf, or TeamNumber\_final.pdf containing a PDF of your writeup.
* For the presentation, you can either publish it online and tell me the link or upload a video.

For TeamNumber\_final.pdf, you should also submit supplementary material. There are two ways to do this. First, you can just package it up:

* **Code:** Upload your code to GitHub/Bitbucket/etc and include a link to the code in the writeup. Alternatively, you can create a code.zip file containing the code, upload it to OneDrive/Google Drive/etc., and include a link in the writeup. Any language is fine; it does not have to run out-of-the-box. You should also include a README file with the repo/zip documenting what everything is and what commands you ran.
* **Data:** Create a data.zip file containing the data. Upload the file to OneDrive/Google Drive/etc. Include a in the writeup.

The file size limit is **20MB** per file. If the data does not fit in the file size limit, submit a small but meaningful subset of the data.

**Grading rubric** Your project will be graded on the following dimensions:

* **Task definition**: is the task precisely defined and is the motivation for the task clear? In other words, does the world somehow become a better place if your project were successful? We will reward projects that are extra thoughtful about how to use **AI for social impact**.
* **Approach**: was a baseline, an oracle, and an advanced method described clearly, well justified, and tested?
* **Data and experiments**: have you explained the data clearly, performed systematic experiments, and reported concrete results?
* **Analysis**: did you interpret the results and try to explain why things worked (or didn't work) the way they did? Do you show concrete examples?
* **Extra credit**: does the project present interesting and novel ideas (i.e., would this be publishable at a good conference)?

Of course, the experiments may not always be successful. Getting negative results is normal, and as long as you make a reasonably well-motivated attempt and you explained *why* the results came out negative, you will get credit.

**An example strategy**

This is a suggestion of how to approach the final project with an example.

* Pick a topic that you're passionate about (e.g., food, language, energy, politics, sports, card games, robotics). *As a running example, say we're interested in how people read the news to get their information.*
* Brainstorm to find some tasks on that topic: ask "wouldn't it be nice to have a system that does X?" or "wouldn't it be nice to understand X?" A good task should not be too easy (sorting a list of numbers) and not too hard (building a system that can automatically solve CS221 homeworks). Please come to office hours for feedback on finding the right balance. *Let's focus on recommending news to people.*
* Define the task you're trying to solve clearly and convince yourself (and a few friends) that it's important/interesting. Also state your evaluation metric – how will you know if you have succeeded or not? *Concentrate on a small set of popular news sites: nytimes.com, slashdot.org, sfgate.com, onion.com, etc. For each user and each day, assume we have acquired a set of articles that the user is interested in reading (training data). Our task is to predict for a new day, given the full set of articles, the best subset to show the user; evaluation metric would be prediction accuracy.*
* Gather and clean the necessary data (this might involve scraping websites, filtering outliers, etc.). This step can often take an annoyingly large amount of time if you're not careful, so do not try to get bogged down here. Simplify the task or focus on a subset of the data if necessary. You might find yourself adjusting the task you're trying to solve based on new empirical insights you get by looking at the data. Notice that even if you're not doing machine learning, it's necessary to have data for evaluation purposes. *Write some scripts that download the RSS feeds from the news sites, run some basic NLP processing (e.g., tokenization), say, using NLTK.*
* Implement a baseline algorithm. For a classification task, this would be always predicting the most common label. If your baseline is too high, then your task is probably too easy. *One baseline is to always produce the first document from each news site.* Also implement an oracle, for example, *recommending the document based on the number of comments. This is an oracle because you wouldn't have the number of comments at the time you actually wanted to recommend the article!*
* Formulate a model and implement the algorithm for that model. You should try several variants and compare them. Remember to try as much as possible to separate model (what you want to compute) from algorithms (how you do it). *You might train a classifier to predict, for each news article, whether to include it or not. You might try to include these predictions as factors in a weighted CSP and try to find a set of articles that balance diversity and relevance.*
* Perhaps the most important part of the project is the final step, which is to analyze the results. It's more important that you do a thorough analysis and interpret your results rather than implement a huge number of complicated heuristics in trying to eke out the maximum performance. The analysis should begin with basic facts, e.g., how much time/memory did the algorithm take, how does the accuracy vary with the amount of training data? What are the instances that your system does the worst on? Give concrete examples and try to understand why. Is there a bottleneck? Is it due to lack of training data?

**Datasets**

You are free to use existing datasets, but these might be not necessarily the best match for your problem, in which case you are probably better off making your own dataset.

* [Kaggle](http://www.kaggle.com/) is a website that runs machine learning competitions for predicting for monetary reward.
* [Past CS229 projects](http://cs229.stanford.edu/projects2011.html): examples of machine learning projects that you can look at for inspiration. Of course your project doesn't have to use machine learning – it can draw from other areas of AI.
* [SAT competition](http://baldur.iti.kit.edu/SAT-Challenge-2012): satisfiability problems are a special important class of CSPs.
* [Natural language processing datasets](http://nlp.stanford.edu/links/statnlp.html#Corpora): links to many NLP datasets for different languages.

**Libraries**

You are free to use existing tools for parts of your project as long as you're clear what you used. When you use existing tools, the expectation is that you will do more on other dimensions.

* [scikit-learn](http://scikit-learn.org/): machine learning library implemented in Python
* [Natural language Toolkit (NLTK)](http://nltk.org/): a set of tools for basic NLP in Python
* [OpenCV](http://www.neuroforge.co.uk/index.php/getting-started-with-python-a-opencv): Python libraries for simple computer vision

**Some project ideas**

* Predict the price of airline ticket prices given day, time, location, etc.
* Predict the amount of electricity consumed over the course of a day.
* Predict whether the phone should be switched off / silenced based on sensor readings from your smartphone.
* Auto-complete code when you're programming.
* Answer natural language questions for a restricted domain (e.g., movies, sports).
* Search for a mathematical theorem based on an expression which normalizes over variable names.
* Find the optimal way to get from one place on Stanford campus to another place, taking into account uncertain travel times due to traffic.
* Solve Sudoku puzzles or crossword puzzles.
* Build an engine to play Go, chess, 2048, poker, etc.
* Break substitution codes based on knowledge of English.
* Automatically generate the harmonization of a melody.
* Generate poetry on a given topic.