

CS 1671 / CS 2071

Human Language Technologies

Session 15: Project proposal presentations

October 13, 2026



University of
Pittsburgh

School of Computing and Information

Schedule

1. Archive of Pittsburgh Language and Speech (APLS)
2. Spanish dialect prediction
3. Book soundscapes
4. News entity prediction
5. Recipe classification
6. Movie summarization
7. Restaurant type classification from reviews
8. Job postings classification
9. Sensitive data detection
10. Portfolio creator
11. Gaming voice chat

Instructions

- Plan for **5 min presentations max** not including Q&A
- Cover at least these key points
 - Project motivation (what is the value of this work?)
 - What data you are planning to use
 - What baseline approach/methods you plan to take
 - How you will evaluate your approach
- Put your slides in this presentation after your project name slide by **class session, 1pm on Mon Mar 10**

1. Archive of Pittsburgh Language and Speech (APLS)

CS 1671 Project

APLS POS tagging group

Project

The goal of the project is to create an NLP model that takes Voice Recordings and their transcriptions from the APLS database and Parts-of-Speech tags them.

APLS (Archive of Pittsburgh Language and Speech): Free corpus of interviews and transcripts with native Pittsburgh speakers

This is an ongoing project led by Prof. Dan Villarreal

These are our names



Hsien Zo



Kang Shin Young



Amelia



Aaron

The Data

The APLS data:

Recordings of interviews

Transcripts of interviews, formatted in "turns"

Automatic POS tags

Currently generated by the Stanford POS tagger, (english-bidirectional-distsim)

We also have access to a hand-tagged set of roughly 3K+ tokens of white speakers and 1K+ tokens of black speakers. This will be used as our test set.

Project Motivation

The same word can mean different things depending on its grammatical role

Ex. “I **record** music” (verb) vs. “a **record** of it” (noun)

Grammatical and semantic analysis of thousands of words depends on accurate tags

Taggers trained on standard written English mistag African American English at higher rates, which skews linguistic research on those speakers

APLS supports Prof. Villarreal’s ongoing research on the Pittsburgh dialect, so the tags need to be reliable for his analyses

Approach/Method

Evaluate baseline tagging model (Stanford) against our hand-tagged set

Build or tune a BERT based model with improvements, keeping in mind:

- contractions
- filler words
- POS differing in spoken vs written language

Explore LLM based approaches

Evaluation

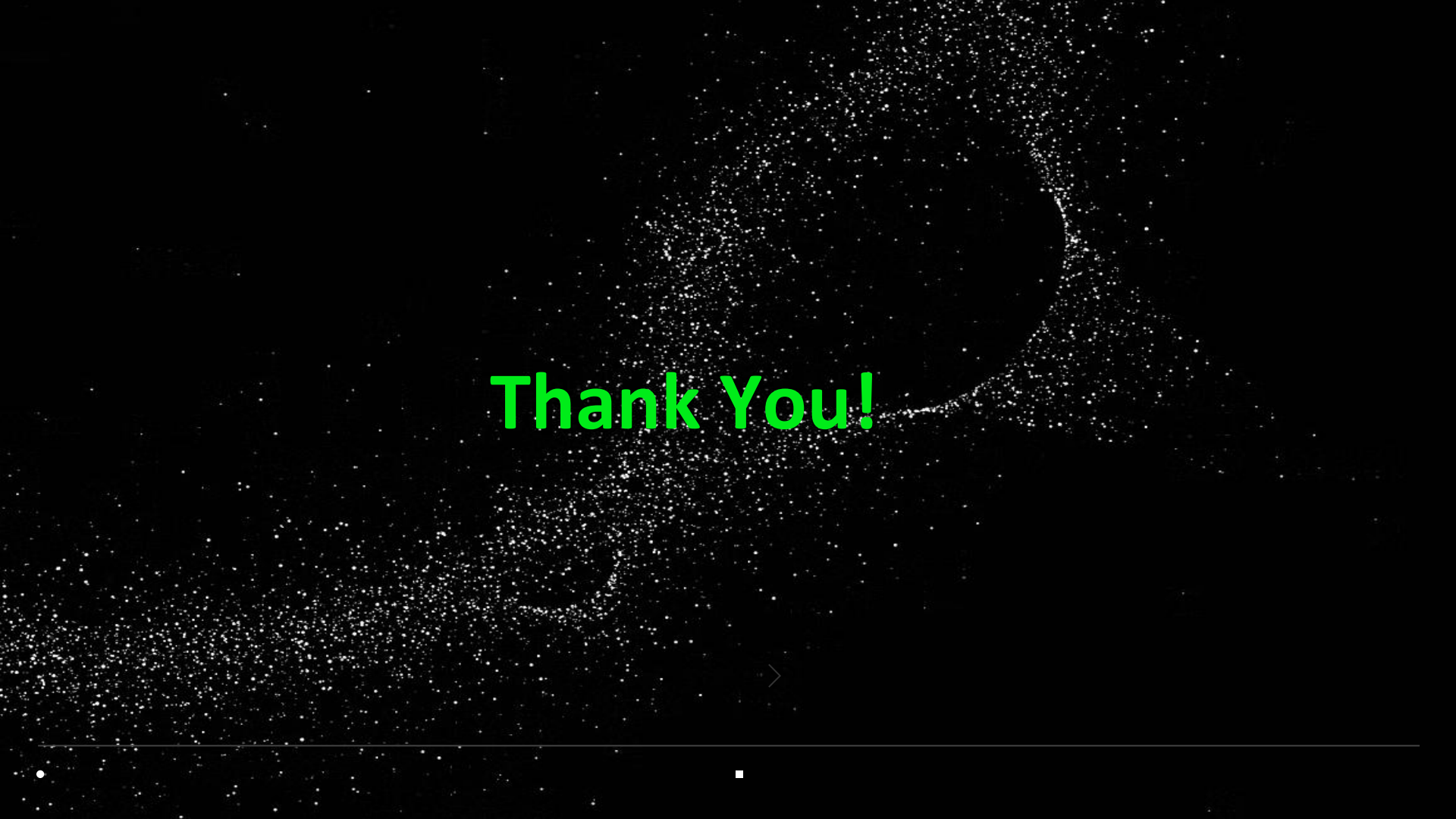
Primarily we will evaluate against our test set, token-level accuracy

In addition, there are previous models that we can compare our methods against

We will also focus on differences in accuracy and other metrics based on speaker, and parts of speech.

Error analysis will rely on confusion matrices, and other metrics to determine the category of errors: disfluency, ambiguity, etc.

Expected metrics: Accuracy, F1 score, precision and recall



Thank You!



2. Spanish dialect prediction

SPANISH DIALECT PREDICTION

PROJECT MOTIVATION

- Use Spanish/French speech to train a model to predict a speaker's dialect
- Spanish and French are both spoken around the world in vastly different dialects
- Differences in pitch, tone, speed, vocabulary and grammar in dialects
- Can we identify a speaker's background based on phonetic features?

DATA

- Audio clips of Spanish and French human speech
- 5-15 second clips, clean audio with no background noise, ~350 per country, both male and female speakers from each
- Spanish: Spain, Mexico, Argentina, Chile, Dominican Republic
- French: France, Belgium, Canada, Ivory Coast, Senegal
- Limitation: Data is not from **all** French or Spanish speaking countries
- Limitation: Speech is from podcasts, somewhat monotonous compared to natural language

METHODS

- Feature extraction
 - Using a pretrained model to recognize phones
 - Determine prosodic features using Praat (Pitch, Speech Rate, Vowel Sounds)
- We'll use these to train a classifier from Scikit-learn
- If we can't get the audio to work out, we'll instead use the transcript
 - Vocabulary and morphology as features (e.g. regional word choices)

Audio clip → Features (phones + Praat) → Classifier → Country

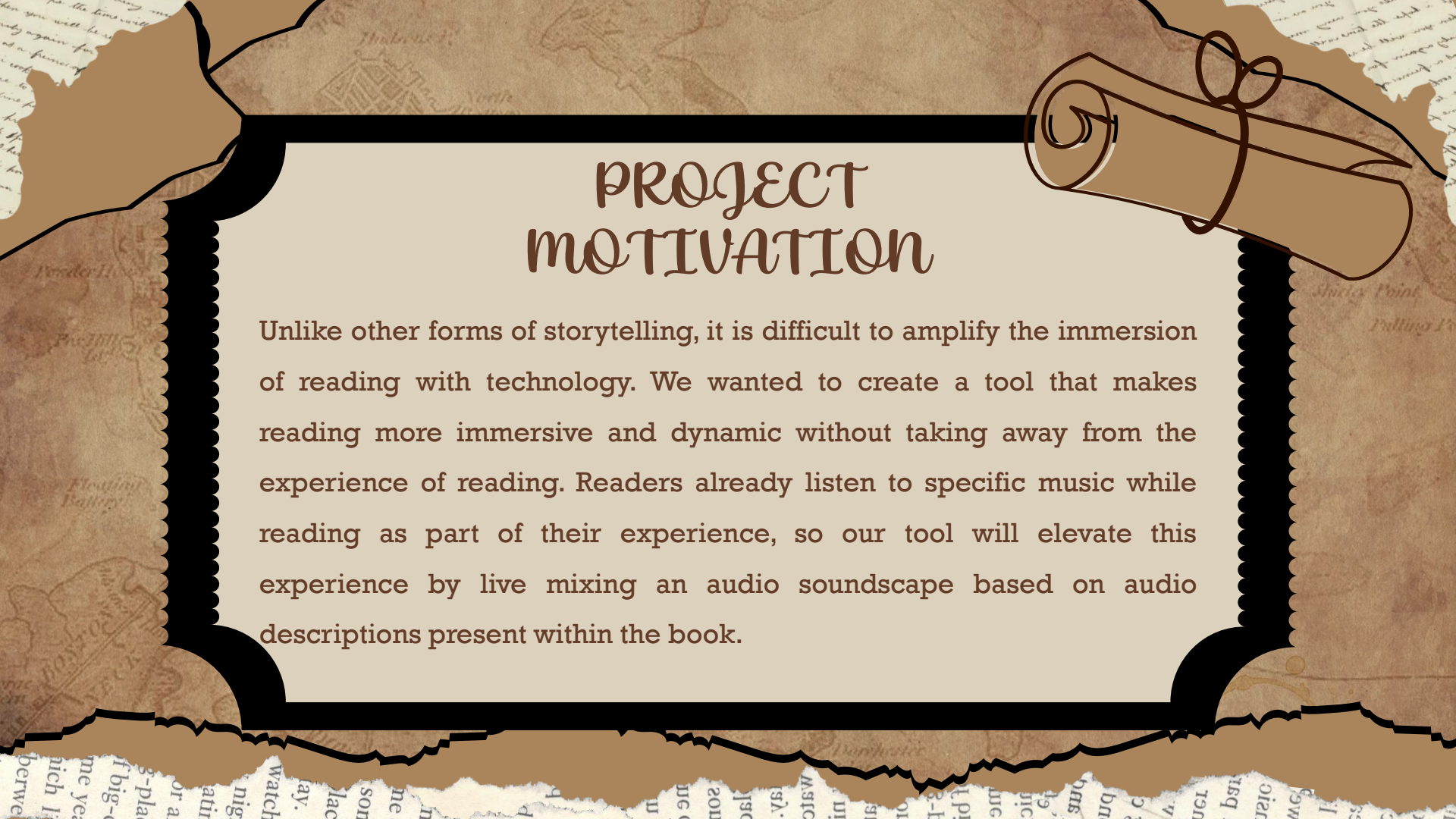
EVALUATION

- Supervised learning (pre-labeled) -> Split into a training and testing set
- F1 scores and Confusion Matrices
 - Checking how often the model managed to guess the right country
- Checking which countries were frequently mistaken for each other.

3. Book soundscapes

The background is a detailed vintage map of Boston Harbor, showing landmarks like the North End, South End, and various wharves. A compass rose is positioned in the upper left. The entire scene is framed by a torn, brown paper border. In the bottom right corner, there is an illustration of a hand holding an open book with cursive handwriting.

BOOK SOUNDSCAPE



PROJECT MOTIVATION

Unlike other forms of storytelling, it is difficult to amplify the immersion of reading with technology. We wanted to create a tool that makes reading more immersive and dynamic without taking away from the experience of reading. Readers already listen to specific music while reading as part of their experience, so our tool will elevate this experience by live mixing an audio soundscape based on audio descriptions present within the book.

DATA

We will be using AudioSet, a dataset developed by Google consisting of human-labeled YouTube clips annotated with sound events. There are 22,176 audio clips across 526 categories. Using the annotations, each clip can be matched with portions of text within a book, a process that links keywords in the narrative (for example "thunder," "footsteps," or "church bell") to the sound class that best represents them.

APPROACH

Essentially a matching problem!

For our first approach, we turn each sentence and each audio caption into a vector, then pick the caption with the highest cosine similarity. Then, we add a threshold, where if a sentence's best similar sound is below the threshold, a sentence gets no sound, and above, it gets its most similar sound. This avoids audio pollution.

We also plan on training a sound/no-sound classifier on our own labels.





EVALUATION

For evaluation, we will score the generated soundscape against our gold text dataset. The gold dataset will be a full, manually annotated copy of “The Hobbit,” which will be created by one of our group members. We will also do additional testing with other texts beyond “The Hobbit” by manually scoring the generated soundscape on a scale based on its perceived accuracy and efficacy.

THANK
YOU



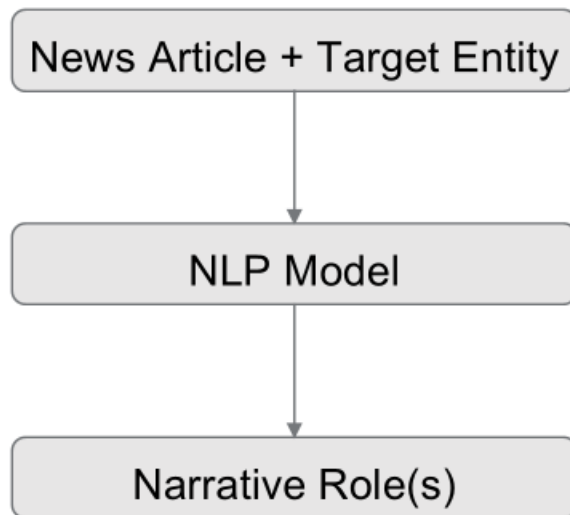
4. News entity prediction

Entity Framing in Online News

Why does this matter?

- News articles do more than report events — they also frame people and organizations in particular roles.
- The same entity can be portrayed as a protagonist, antagonist, or innocent party, depending on the article.
- Automatically identifying these roles can help analyze narratives and potential manipulation in online news.

Task



Goal: Predict how an entity is portrayed in the article

What Data Will We Use?

Data Source

- SemEval 2025 Task 10: Entity Framing in News
- Official task dataset (English, Subtask 1)
- News articles and entity annotations

Our Processed Data

- 202 unique annotated news articles
- 686 annotated entity mentions

What Each Sample Contains

- Article ID and entity name
- Entity position in the article (character offsets)
- Main role: Antagonist / Protagonist / Innocent
- Fine-grained role labels (multiple labels possible)

EXAMPLE DATA RECORD

Article ID	EN_CC_100013.txt
Entity	Bill Gates
Character offsets	93–102
Main role	Antagonist
Fine-grained roles	Deceiver, Corrupt



Method: Feature-Based Statistical NLP

Input Preparation

- Treat each entity mention as one data point.
- Extract its surrounding context using character offsets.

Feature Extraction

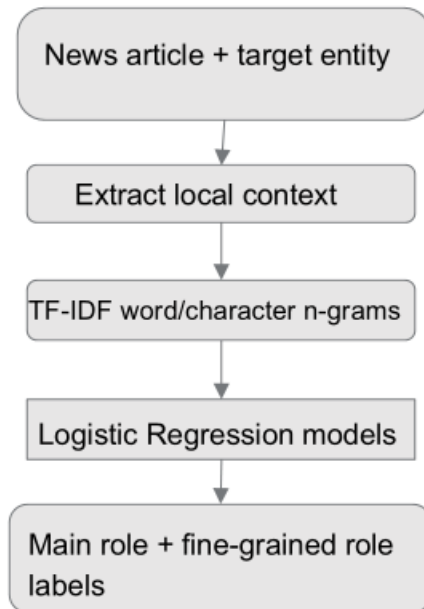
- Represent the context with TF-IDF word n-grams.
- Compare unigrams, bigrams, and different context sizes.

Classification

- Use multiclass Logistic Regression for main roles.
- Use One-vs-Rest Logistic Regression for fine-grained roles.

Model Selection

- Use the development set to tune features, context size, and model parameters.



Evaluation

Exact Match Ratio (Primary)

Full predicted fine-grained label set must match the gold labels.

Micro-F1 & Macro-F1

- Micro-F1 measures overall performance across all label predictions.
- Macro-F1 gives rare and frequent roles equal weight.

Example: Gold = {Deceiver, Corrupt}, Prediction = {Deceiver}

Exact Match gives no credit, while F1 gives partial credit for correctly predicting “Deceiver.”

Model Comparison

Compare unigrams, unigrams + bigrams, and context-window sizes using the official English train/dev split.

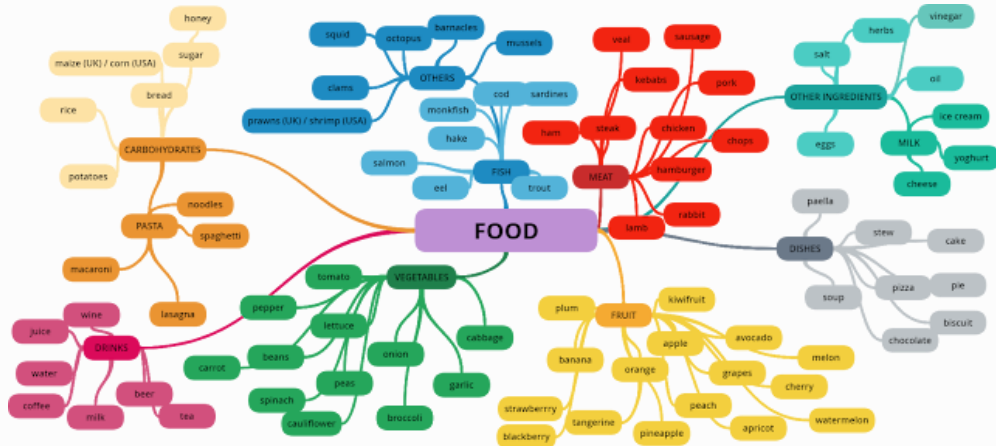
Error Analysis

Examine confused roles, rare-label errors, and missing or extra predictions.

5. Recipe classification

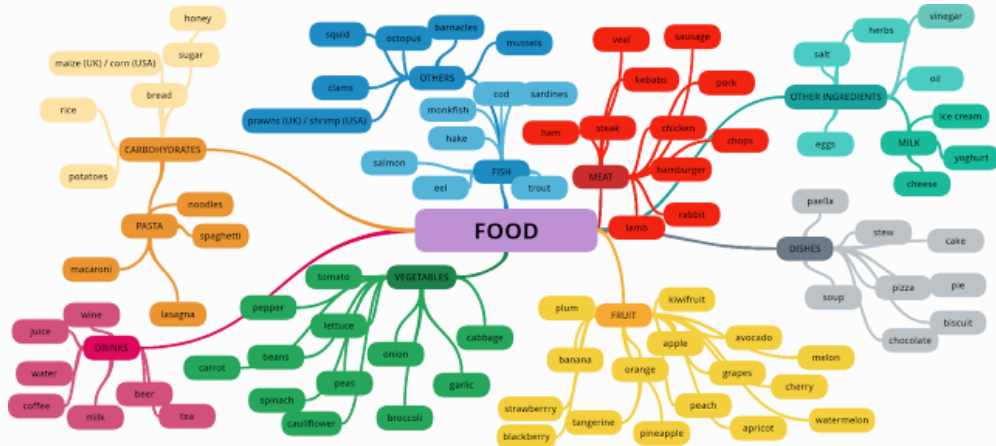
Project Intro

- We want to classify recipes into cuisines based on their ingredients



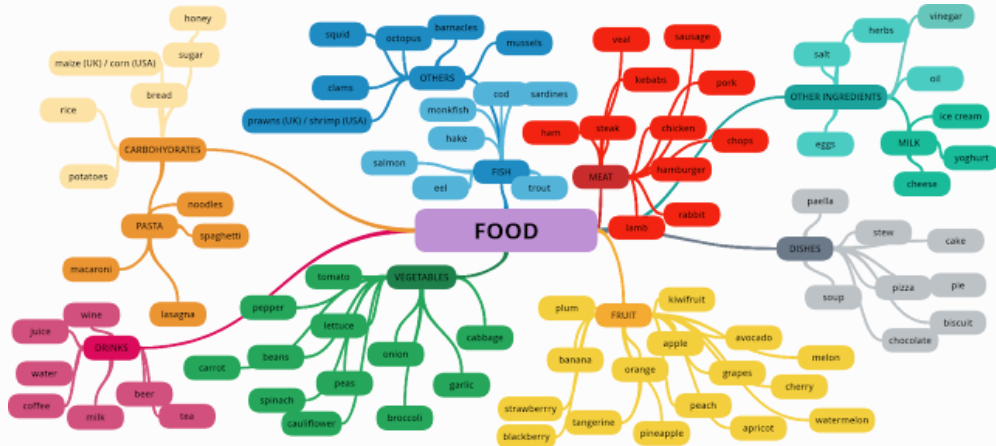
Project Intro

- We want to classify recipes into cuisines based on their ingredients



Project Intro

- We want to classify recipes into cuisines based on their ingredients



Project Motivation

- We want to analyze the similarity between recipes and see the overlap of ingredients to potentially try new cuisines.
- We can see ingredient trends in different cultures (i.e. the most unique and universal ingredients).



Data

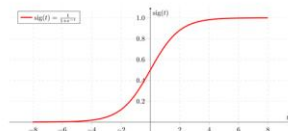
- Yummly data from the "What's Cooking?" Kaggle competition
 - Yummly went bankrupt and we can't look at the source website
 - Recipes already classified into 20+ cuisines



Approach

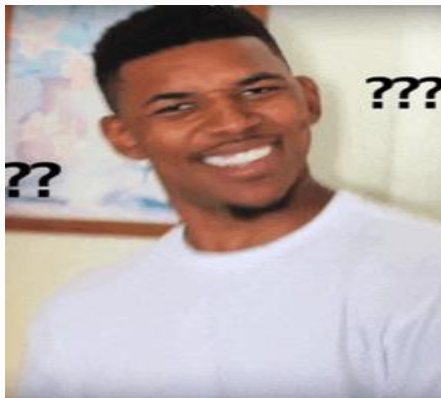
- Preprocessing approaches:
 - Stemming
 - Tomatoes to tomato
 - Eggs to egg
 - Olive oil, vegetable oil to oil
- Unigram tokenization
 - (each ingredient = token)
- Multinomial logistic regression

Logistic Regression
be like:



Evaluation

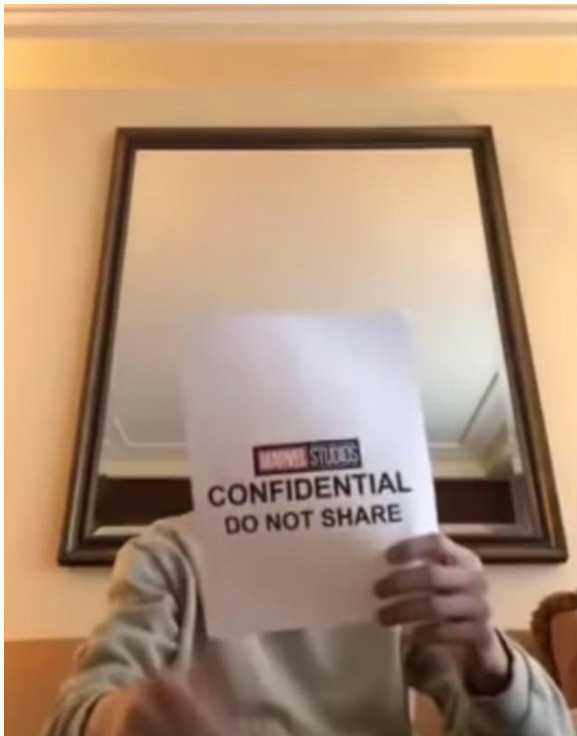
- Confusion matrix for each cuisine
 - Accuracy, Precision, and Recall
- Pooled confusion matrix and microaveraged and macroaveraged precision



6. Movie summarization

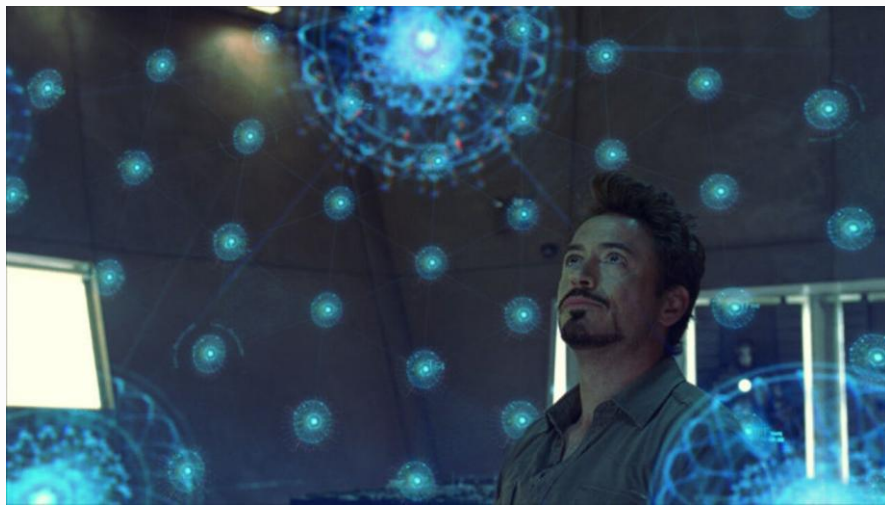
Project motivation (what is the value of this work?)

- The MCU has 100+ connected titles, so viewers need quick recaps instead of rewatching
- Subtitles are a cheap, text-only substitute for video
- Applications include auto-recaps, plot Q&A
- Shows how far classical NLP gets compared to LLMs



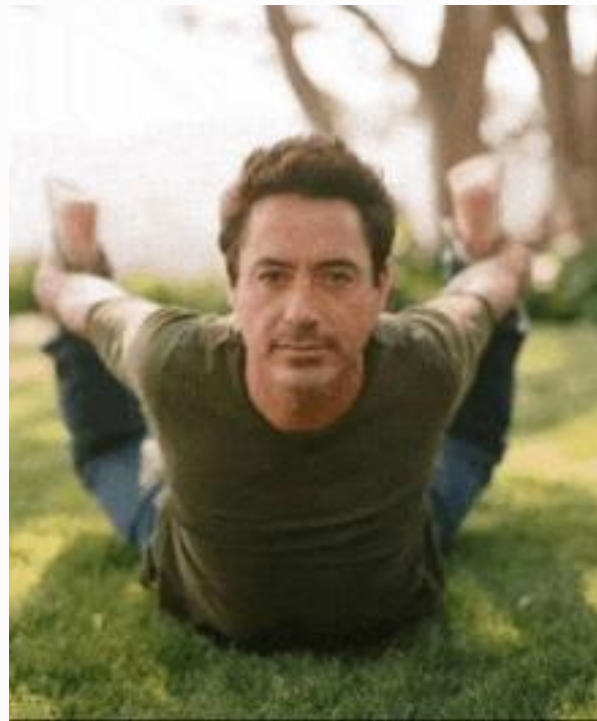
What data you are planning to use

- Subtitles to summarize each Marvel movie/TV episode
- 100+ subtitles from OpenSubtitles.com, a large database (over 3.5 M subtitles across 51 languages) with API
- Gold Summaries from Wikipedia paired with each transcript to compare



What baseline approach/methods you plan to take

- Summarization
 - TF-IDF on unigrams and bigrams
 - Each movie/episode is one document
 - Scoring each subtitle line: (sum of term weights)/(length)
 - Top-scoring lines in original order form the summary
 - spaCy for tokenizing, POS tagging, and boosting proper nouns (ex. character names)
- Question answering
 - Match questions to TF-IDF similarity
- LLM methods



How you will evaluate your approach

- We will evaluate the summaries by using the ROUGE approach
 - Comparing each generated summary against reference summaries
- All systems will produce summaries of the same length for a fair comparison
 - Matching TF-IDF system against a lead baseline and a random baseline, and compare unigram and bigram
- Question answering
 - Create our own question set and score answers with exact match and token-level F1



7. Restaurant type classification from reviews

Project Motivation



Restaurant type classification

- Improve restaurant categorization
 - Automatically classify restaurants by type of food using customer reviews
- Extract insights from text
 - NLP to identify food-related keywords and language patterns that indicate different cuisines
- Reduce manual effort
 - Automate restaurant classification instead of relying on people to label restaurants manually
- Explore real-world NLP applications
 - Show how machine learning can turn unstructured customer reviews into useful information

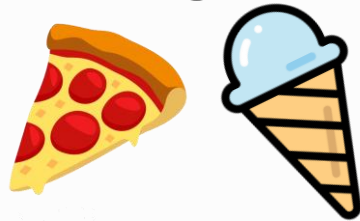
What data you are planning to use



- The Big Yelp Open Data Set:
- <https://business.yelp.com/data/resources/open-dataset/>
- 8.65 GB of uncompressed JSON
- Took 2 of the JSONS, business.json and review.json and combined them into a CSV
- Restaurant format: review_id, business_id, business_name, city, state, stars, date, text
- We have about 4,724,471 restaurant reviews across 52,268 businesses

Methods

- Restaurant type prediction as a text-classification problem.
- Represent the text as features using n-grams (unigram, bigrams, trigrams, etc.)
- Examples are "pizza shop", "ice cream".
- Represent these features using their counts in the reviews.
- Using Multinomial Logistic Regression for predicting from multiple categories.



Evaluations

- Evaluate using Accuracy, Precision, Recall, and F1 Score
 - Accuracy - how many predictions are correct overall
 - Precision - how often a predicted category is actually correct
 - Recall - how many of the actual examples in a category we successfully find
 - F1 score combines precision and recall into one measure, for example, if our model is very precise but misses many actual American restaurants, the F1 score will reflect that problem.
- Use Macro F1 because restaurant categories are imbalanced
 - Seafood ~400,000 reviews
 - Fast food ~30,000 reviews
 - Macro F1 gives each category equal importance
- Compare unigram, bigram, and trigram feature combinations
 - Which one gives us the best performance
- Check performance for each restaurant category
 - To see which restaurant types are easier or harder to classify

8. Job postings classification

9. Sensitive data detection

Dante, Kyle, Sharvani, Chris, YiPing

Project Motivation

- Everyday personal identifiable information (PII) is leaked and used for nefarious purposes.
- We want to look into ways in which this information can be safeguarded using LLMs.
- We hope that the findings from the project give us a better sense of how LLMs can help protect people's data.

Data

- Nemontron-PII (NVIDIA, Hugging Face): 100k rows of PII data. Synthetic, personal texts.
- Span-level labels for 55 PII/PHI types, converted to token level labels
- Add a small set of our own non-PII texts
- 70k train / 20k dev / 10k test

Approach & Method

- Token-level classifier where each token is assigned a label of PII category (i.e. DOB, phone number, none)
- Text broken into unigrams, bigrams, and character n-grams (for pattern detection common in emails, phone numbers, etc.), comparing which approach is best
- Weights based on term frequency-inverse document frequency
- Multinomial logistic regression model to predict the various PII categories

Evaluation

- **Token-level Accuracy:** Measure the proportion of correctly classified tokens.
- **Precision, Recall, and F1 Score:** Evaluate PII detection performance, with particular attention to missed sensitive information (false negatives).
- **Per-class Evaluation & Macro-F1:** Compare performance across PII categories (e.g. DOB, phone numbers, SSNs), accounting for class imbalance.
- **Error Analysis:** Examine false positives and false negatives to identify common misclassification patterns.

Thanks, any Questions?

10. Portfolio creator

PROJECT PROPOSAL

Portfolio Generator

Turning computer science resumes into portfolio-ready content with NLP

Alyssa Signorelli · Aiden McCoy · Aidan Krieger · Sharvi Arora

Why build a portfolio generator

**A resume already
holds everything a
portfolio needs.**

1

Resumes are dense

One page of short, technical bullet points that are hard to skim.

2

Portfolios take effort

Rewriting the same experience, projects and skills for a website is slow and manual.

3

Automate the bridge

Extract resume sections with NLP and feed them straight into a portfolio template.

Paired CS resumes and public portfolios

10

resume + portfolio pairs
for our first pass

Source

Portfolios from the public developer-portfolios GitHub repo (2,000+ sites), plus our own and other public resumes

Why CS only

Keeps resume formatting consistent; other majors use different layouts

What we extract

Section labels: education, experience, projects, skills, summary.
No pre-annotated gold labels.

Extract, normalize, then fill the template



Contact fields (name, phone, email, URLs)
extracted separately

Each entry keeps its bullets as a description the
template uses as is

Flexible schema handles optional sections
(volunteer, certifications)

Optional: LLM summarizer rewrites terse bullets into
portfolio language

```
{
  "name": "John Doe",
  "email": "abc@pitt.edu",
  "Experience": [{
    "Company": "Google",
    "Position": "SWE Intern",
    "Description": "Worked on..."
  }],
  "Projects": [...],
  "Skills": [...]
}
```

Human ratings plus automatic metrics

1

Accuracy

Only facts that appear on the resume. No invented metrics or details.

2

Completeness

Every resume section is represented; scored as a percentage of points kept.

3

Readability & format

Professional tone, little jargon, follows our schema, not overcrowded.

Automatic metrics: ROUGE and BERTScore compare the generated portfolio text against the source resume

11. Gaming voice chat

Project motivation – What are we fixing

- Toxicity in game voice is hard to study: Hateful speech harms a lot of people, but it's spoken, not typed, so we can't analyze it until it's accurately transcribed.
- Transcriptions breaks down in livestreams: Things like background music, game audio, and overlapping voices push word error rate up to 40%
- Better transcripts can unlock everything: Reliable words are the foundation for detecting toxicity, understanding how it spreads, and eventually building safer gaming spaces.



What data you are planning to use

- 485 Livestream Audio (Kick)
 - 11 of the most popular games
 - English language, 18+. 30mins –5 hours
- Livestream transcriptions
- Face-blurred video



What approach/methods you plan to take

- Remove background music:
 - "Cinematic" Audio source separator
 - PyTorch, soundfile, scipy, demucs, NumPy
- Overlapping voices:
 - Audio Source Separation
 - Open-source libraries like SpeechBrain from PyTorch, WhisperX, or Audacity
 - OpenVINO - Audacity Plugin for separating overlapping audio



How will you evaluate your approach?

- Accuracy of transcriptions vs. gold labels
- Word Error Rate: reduce from ~40% to < 20%
 - Compare WER before and after each fix
 - Break down by clip type: background music, game audio, overlapping voices
 - Toxic word recall

