Overview

Linguine aims to allow language science students to explore automated language analyses such as syntactic or semantic analyses.

This project built upon prior senior projects that created a user interface and web API. This year's project focused on linguistic analysis functionalities as well as system stability and performance.

Contributions

- Increased functionality with 5 new analysis types
- Added visualizations for all analysis types
- Implemented concurrency on Python backend
- Added visualizations for all analysis types
- Increased functionality with 5 new analysis types

Technologies

- AngularJS UI
- N-tsc CoreNLP
- NodeJS API
- Python Analysis API
- MongoDB Instance

Analyses & Visualizations

- Term Frequency Analysis: Compute word frequencies in a text
- Named Entity Recognition: Identify words by classes such as organization, place, or time expression
- Coreference Resolution: Locate expressions that refer to the same entity in a text
- Relation Extraction: Find relationship triples between words
- Parsing & Part of Speech Tagging: Construct a dependency parse tree and mark words by part of speech
- Sentiment Analysis: Estimate the sentiment of a text along with its sentences and tokens

Methodology

We followed a Scrum methodology, using two week sprints. Several of our team members had experience using Scrum and we were likely to do much of the development independently, so it was the most appropriate choice.

In Fall, we met twice a week and remotely on the weekends. In Spring, we met in person three times a week to increase productivity.

We began each sprint with a sprint planning meeting to assign story points to each of our user stories. We used burndown charts to track our velocity and estimation accuracy.

Lessons Learned

- Consider a distributed model for expensive operations
- Flush out critical system bugs prior to opening up tool for use
- When inheriting a project, budget time for rework or bug fixing
- Sponsor time is valuable – e.g., helped us with domain concepts