Linguine

Open source natural language processing & visualization

Team Pastafarians: Danielle Gonzalez, Justin Peterson, Jeremy Vasta, Keegan Parrotte

Sponsor: Cecilia Ovesdotter Alm
Coach: Larry Kiser | 2015-2016

Software Engineering
Rochester Institute of Technology

R·I·T
College of Liberal Arts

Overview

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

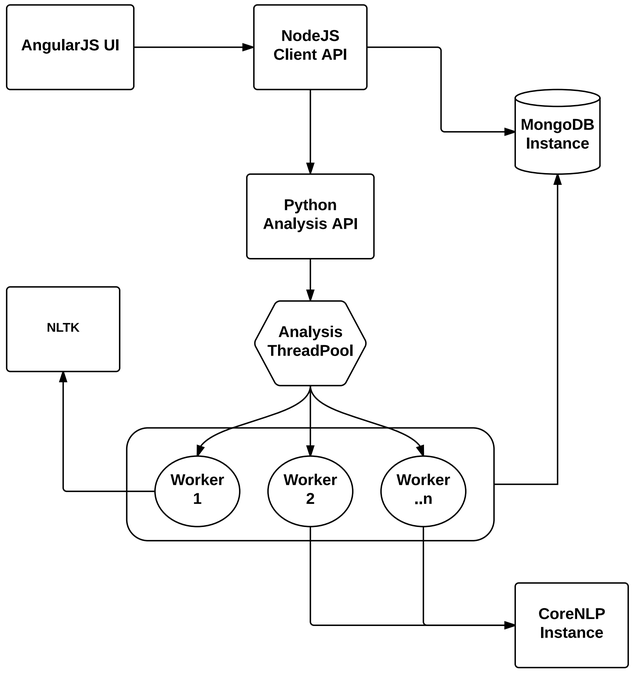
This project built upon prior senior projects that created a user interface and web API. This year's project focused on linguistic analysis functionalities as well as system stability and performance.

Contributions

- Increased functionality with 5 new analysis types
- Added visualizations for all analysis types
- Made multiple usability improvements to UI
- Implemented concurrency on Python backend
- Integrated Stanford CoreNLP and Illinois Curator
- Enabled users to work while analysis is processing

Technologies

Architecture



Python API
Performs analyses with Stanford CoreNLP, NLTK

NodeJS API
Handles front end interactions such as corpus upload and analysis creation

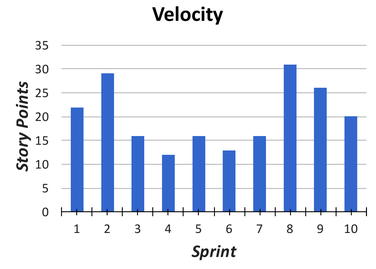
Analysis Thread Pool
Conducts analyses using a group of background server threads

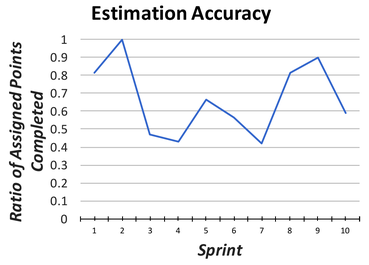
Methodology

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.



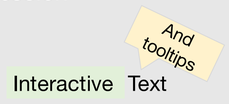


Analyses & Visualizations

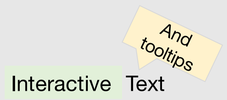
Term Frequency Analysis
Compute word frequencies in a text

Word Cloud

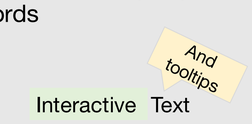
Named Entity Recognition
Identify words by classes such as organization, place, or time expression



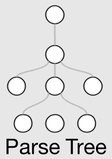
Coreference Resolution
Locate expressions that refer to the same entity in a text



Relation Extraction
Find relationship triples between words

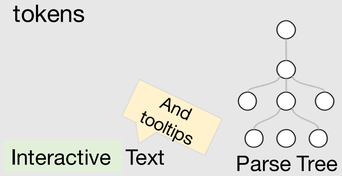


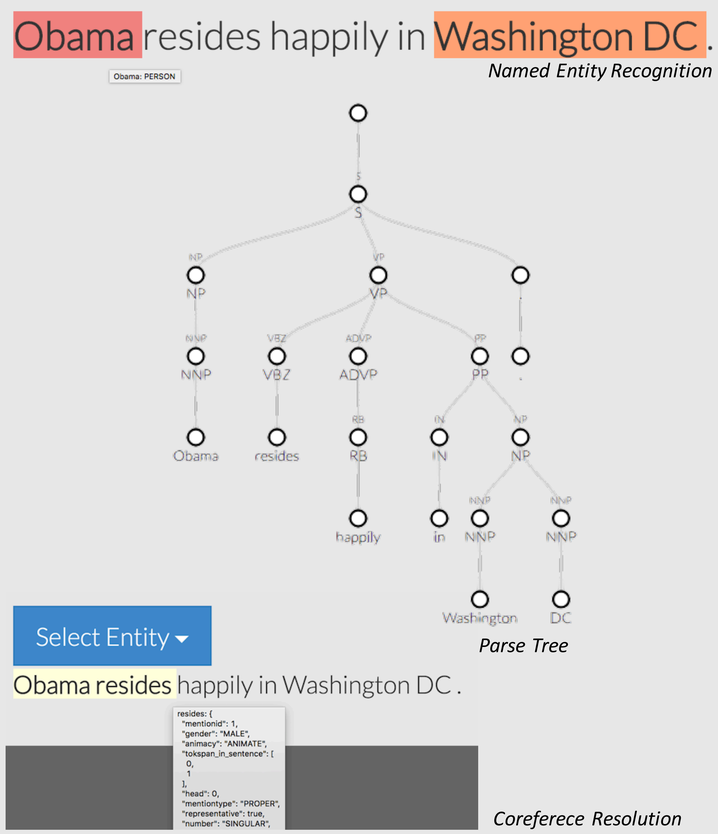
Parsing & Part of Speech Tagging
Construct a dependency parse tree and mark words by part of speech

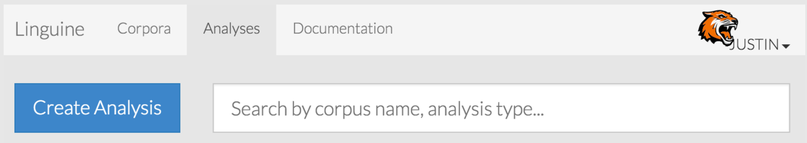


Parse Tree

Sentiment Analysis
Estimate the sentiment of a text along with its sentences and tokens







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