A Part of Speech Tagger for Software Documentation

Problem | Current industry standard part of speech (POS) tagging solutions do not have any means to tag software documentation due to the intermixing of natural language and code.

Solution | Create a POS Tagger that can accept software documentation in the form of HTML files, trained on the back of a mix of automatic and manual tagging of Software documentation.

Joseph Naberhaus | Project Lead
James Taylor | Computational Linguistics SME
Austin Boling | Meeting Facilitator
Ekene Okeke | Report Coordinator
Ahmad Alramahi | Lead Developer
Ethan Ruchotzke | Documentation Manager
sdmay21-35

Faculty Advisor | **Ali Jannesari**
Graduate Supervisor | **Hung Phan**

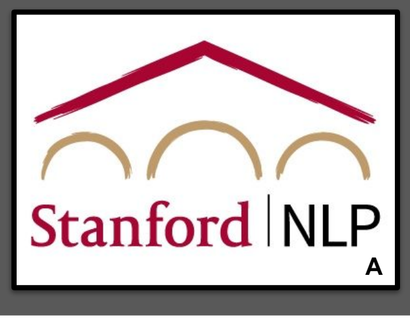
Requirements

Functional

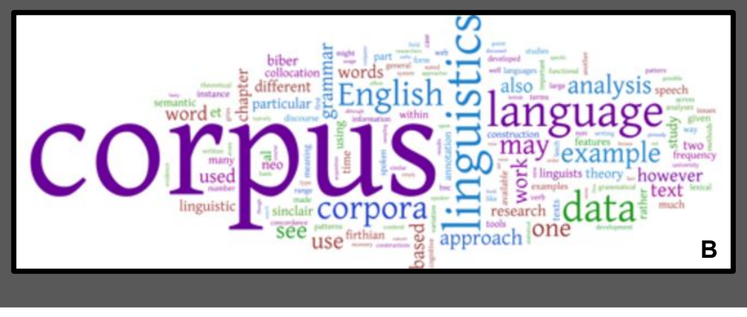
Create a corpus of tagged software documentation
Augment Stanford coreNLP model
Tag software documentation with a greater accuracy than a default English tagger would

Non-Functional

Accessible to an average user of general software packages
Tag thousands of tokens in a matter of seconds, not minutes



A



B

Engineering Constraints

Works within, and adhere to the requirements of, coreNLP
Limited time resources for engineering work

Operating Environment

JDK 15 (or SE 16) & Python 3.8 or above

Intended Users

Researchers, developers, and students looking to combine natural language processing and software documentation.

Standards

ISO-IEC 12207 | Software Life Cycle - Longevity of project
ISO-IEC 9001 | Quality Management - Quality of project
ECMA 494 | JSON - Data transfer in pipeline

Design Approach

Training A New Model

1. Scrape the documentation
2. Parse the documentation
3. Tokenize and sentence split the documentation
4. Automatically tag the documentation, use manual intervention for clean up
5. Train the model

Results in: *trained CRF model for tagging software docs*

Tagging New Software Documentation

1. Parse the new documentation
2. Tokenize and sentence split the documentation
3. Consult model

Results in: *tagged docs*

New Tag Set

40 Original Penn TreeBank Tags [NN = noun, VBZ = verb]
30 New Software POS Tags [<val> = value, <type> = type]
~5 HTML Tags [<code> = HTML code tag]

Technical Details: Modules

HTML Scraper
Python

HTML Parser
Parse HTML Based on a tag whitelist

Using lxml, glob

Python

Tokenization
Tokenizes and sentence splits parsed HTML based on rules

Java

AutoTagging
Automatically tags when confident, manual tagging GUI for clean-up

Using stanford.nlp, javafx

Java

POSModel
Trains a CRF from JSON files, Tags new documentation, and tests accuracy

Using stanford.nlp, fasterxml, apache commons & log4j

Java

Integration Testing | Our project required heavy integration testing as you can see the 5 distinct modules we used above. Ensuring all modules connect fluidly and bug free was very important.

Model Accuracy | We have created custom testing methods to test accuracy regardless of model used. Our current best result is **~55% accuracy** using the base conditional random field training from Stanford coreNLP.

External Image Sources: A: nlp.stanford.edu, B: https://lope.linguistics.ntu.edu.tw/courses/corpusling2015/lectures/01/index.html,