Tale Scanner : Natural Language Processing

Matteo DeMilia, Kyrstn Hall, Christopher Mead, and Tiera Smith
Professor: Dr. Jing Hou
Sponsor: Dr. Darin Woolpert of the University of Vermont

Background

- Linguistic analysis is slow and tedious
- Speech language pathologists(SLP) would benefit from efficient analysis tools
- Natural language processing(NLP) can speed up the analysis process exponentially
- The future of linguistic analysis lies in computed automation and artificial intelligence(AI)

Why Choose Tale Scanner?

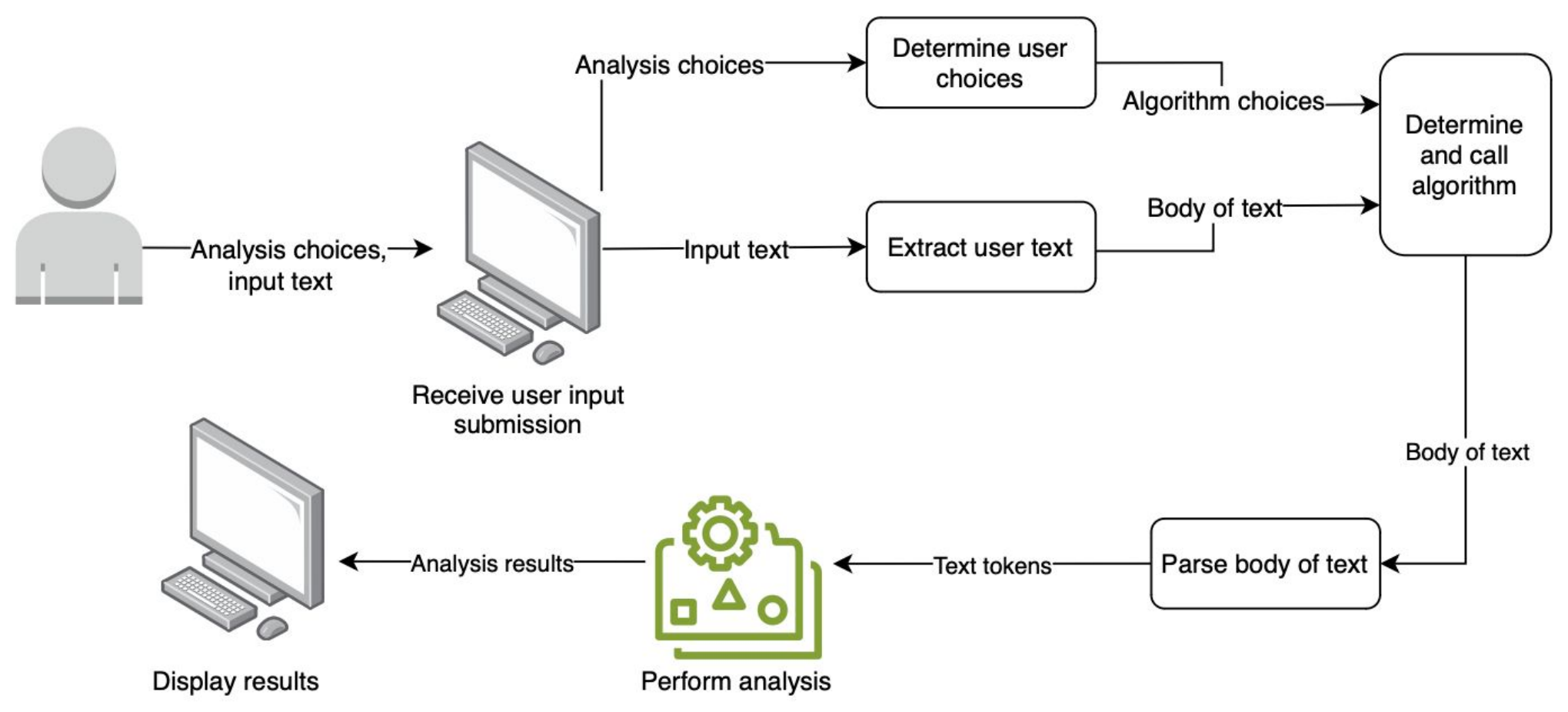
- Tale Scanner is much more efficient than manual linguistic analysis
- Tale Scanner provides consistency, unlike general AI
- Choose from numerous analysis with results provided in <1 min
- Tale Scanner provides detailed description and background on the analysis process of each algorithm

System Requirements

Goals of the Tale Scanner NLP System:

- Provide a simple user interface supporting:
  - Input text-box for 200-600 words
  - Analysis choice selection
  - Seamless execution and navigation
- Provide Text Analysis of:
  - Total number of words and unique words
  - Type token ratio
  - Total number of clauses
  - Total number of subordinate clauses
  - Syntactic subordination index
  - Total number of verb errors
  - Verb error ratio
  - Total number of morphemes

System Architecture and Design

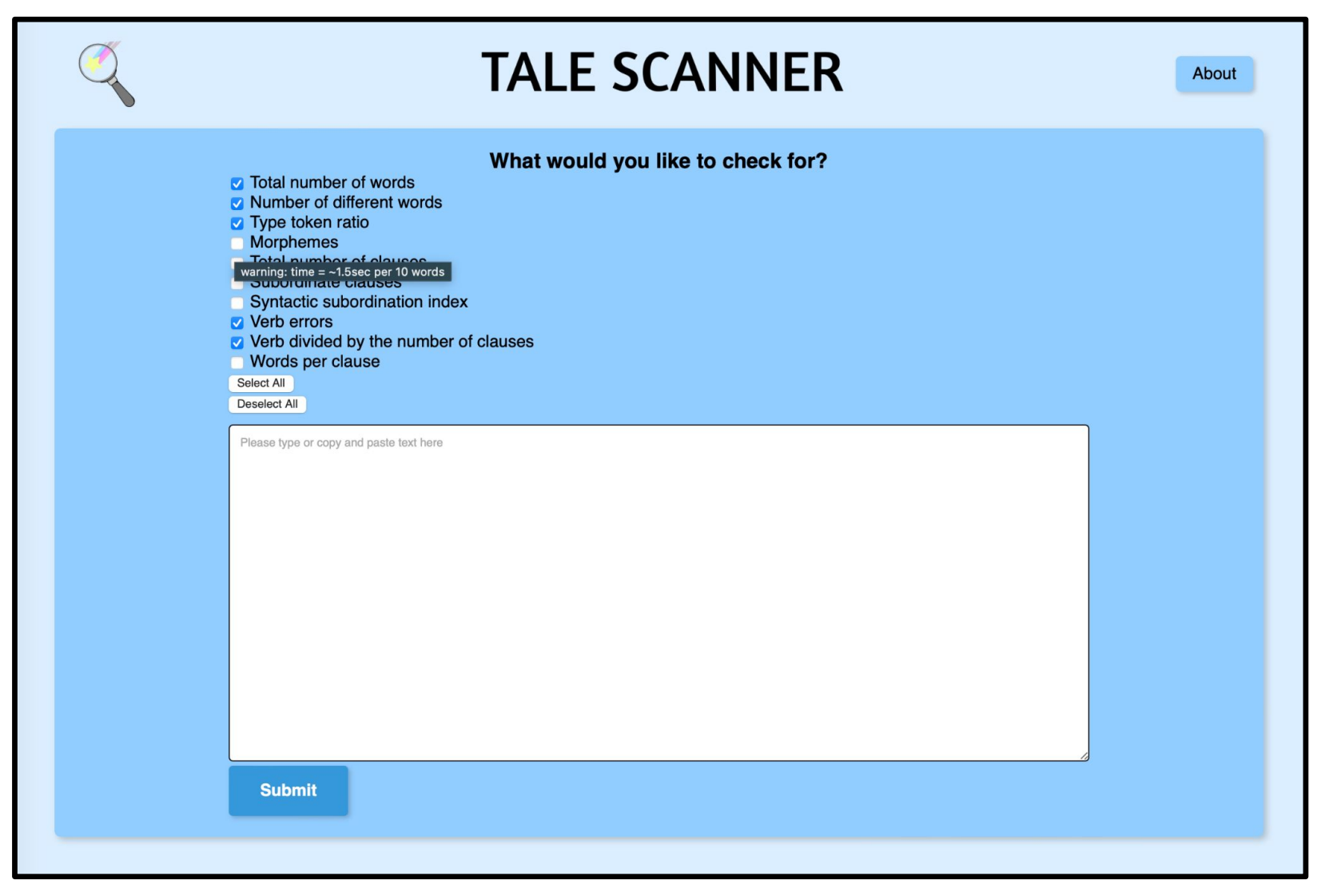


- Simple, linear design; reduces dependencies and error cases
- Designed for efficiency and stability
- Two inputs:
  - Body of text
  - Analysis choices

Tech Stack

HTML CSS JavaScript
Python Flask AWS Amplify

Tale Scanner's User Interface



TALE SCANNER

About

What would you like to check for?

☑ Total number of words
☑ Number of different words
☑ Type token ratio
☐ Morphemes
☐ Total number of clauses
☐ Subordinate clauses
☐ Syntactic subordination index
☑ Verb errors
☑ Verb divided by the number of clauses
☐ Words per clause

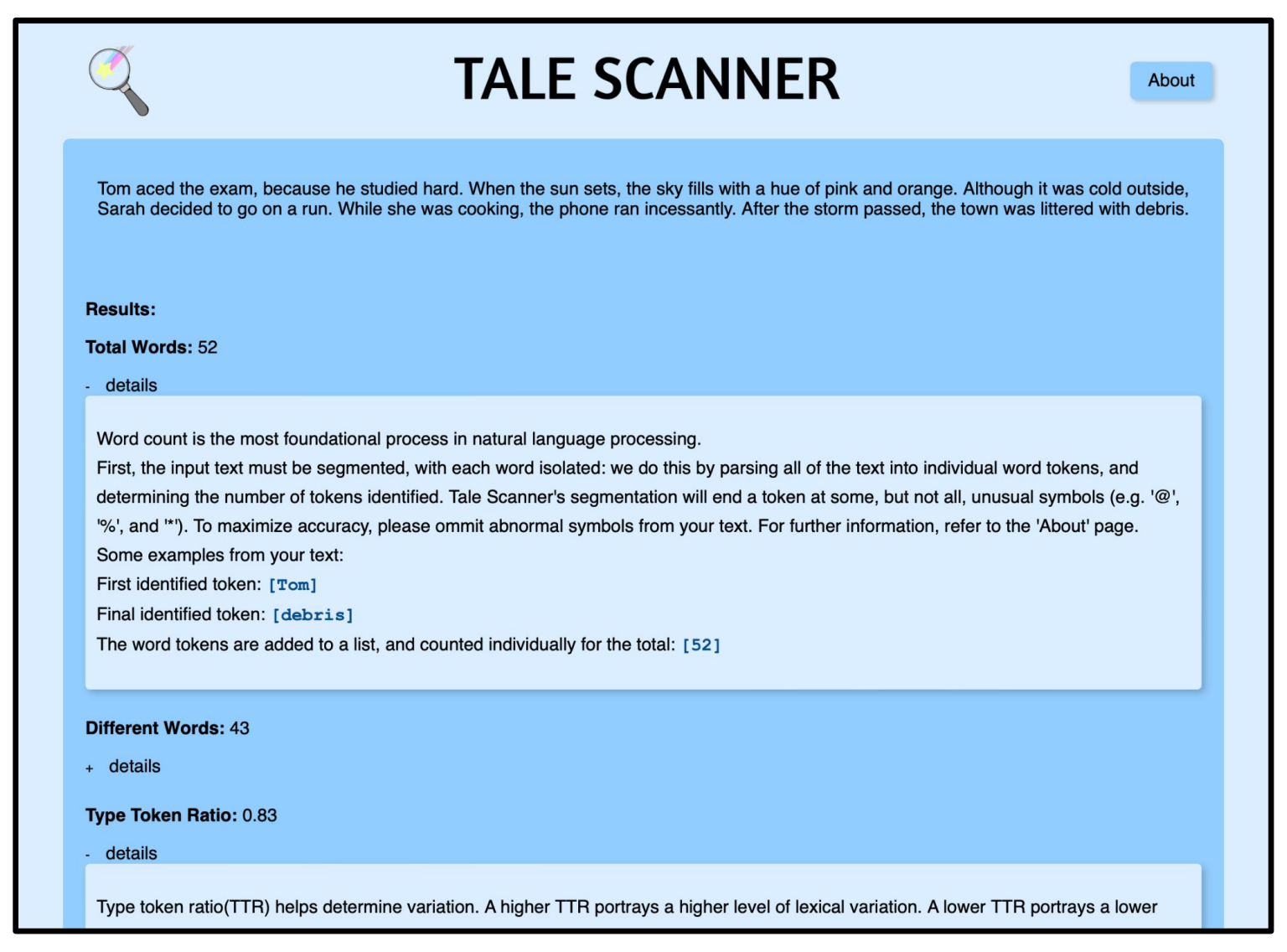
warning: time = ~1.5sec per 10 words

Select All
Deselect All

Please type or copy and paste text here

Submit

- Simple and clean UI
- Spinner submit button
- Error message for input errors
- Saves previously chosen analysis boxes
- Dynamic warnings are provided for a few analysis choices



TALE SCANNER

About

Tom aced the exam, because he studied hard. When the sun sets, the sky fills with a hue of pink and orange. Although it was cold outside, Sarah decided to go on a run. While she was cooking, the phone ran incessantly. After the storm passed, the town was littered with debris.

Results:

Total Words: 52

- details

Word count is the most foundational process in natural language processing.
First, the input text must be segmented, with each word isolated: we do this by parsing all of the text into individual word tokens, and determining the number of tokens identified. Tale Scanner's segmentation will end a token at some, but not all, unusual symbols (e.g. '@', '%', and '*'). To maximize accuracy, please ommit abnormal symbols from your text. For further information, refer to the 'About' page.
Some examples from your text:
First identified token: [Tom]
Final identified token: [debris]
The word tokens are added to a list, and counted individually for the total: [52]

Different Words: 43

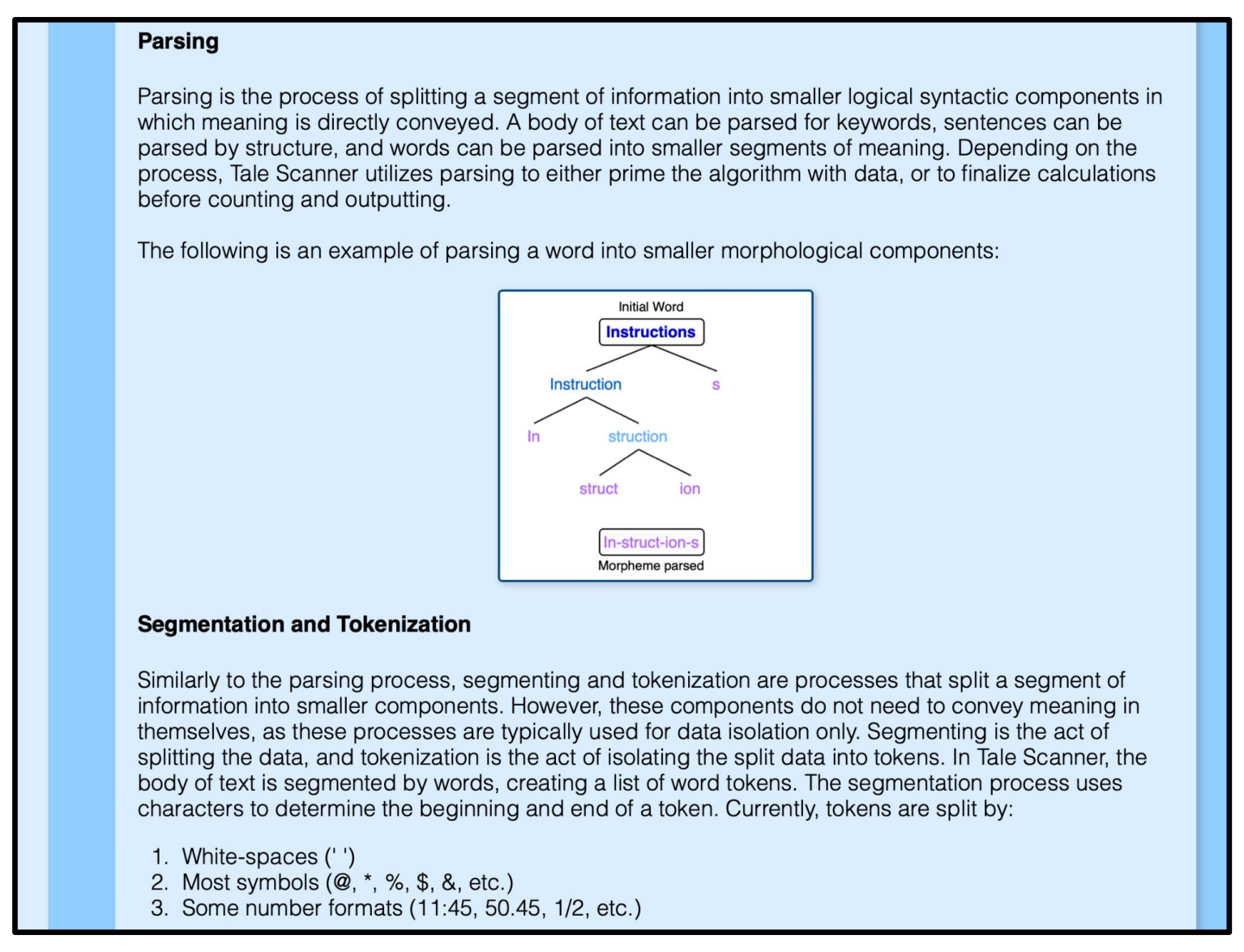
+ details

Type Token Ratio: 0.83

- details

Type token ratio(TTR) helps determine variation. A higher TTR portrays a higher level of lexical variation. A lower TTR portrays a lower

- Results page loads with all chosen analysis results in drop down menu
- Drop down features include:
  - Dynamic data in blue text
  - Insight on the analysis algorithm



Parsing

Parsing is the process of splitting a segment of information into smaller logical syntactic components in which meaning is directly conveyed. A body of text can be parsed for keywords, sentences can be parsed by structure, and words can be parsed into smaller segments of meaning. Depending on the process, Tale Scanner utilizes parsing to either prime the algorithm with data, or to finalize calculations before counting and outputting.

The following is an example of parsing a word into smaller morphological components:

Segmentation and Tokenization

Similarly to the parsing process, segmenting and tokenization are processes that split a segment of information into smaller components. However, these components do not need to convey meaning in themselves, as these processes are typically used for data isolation only. Segmenting is the act of splitting the data, and tokenization is the act of isolating the split data into tokens. In Tale Scanner, the body of text is segmented by words, creating a list of word tokens. The segmentation process uses characters to determine the beginning and end of a token. Currently, tokens are split by:

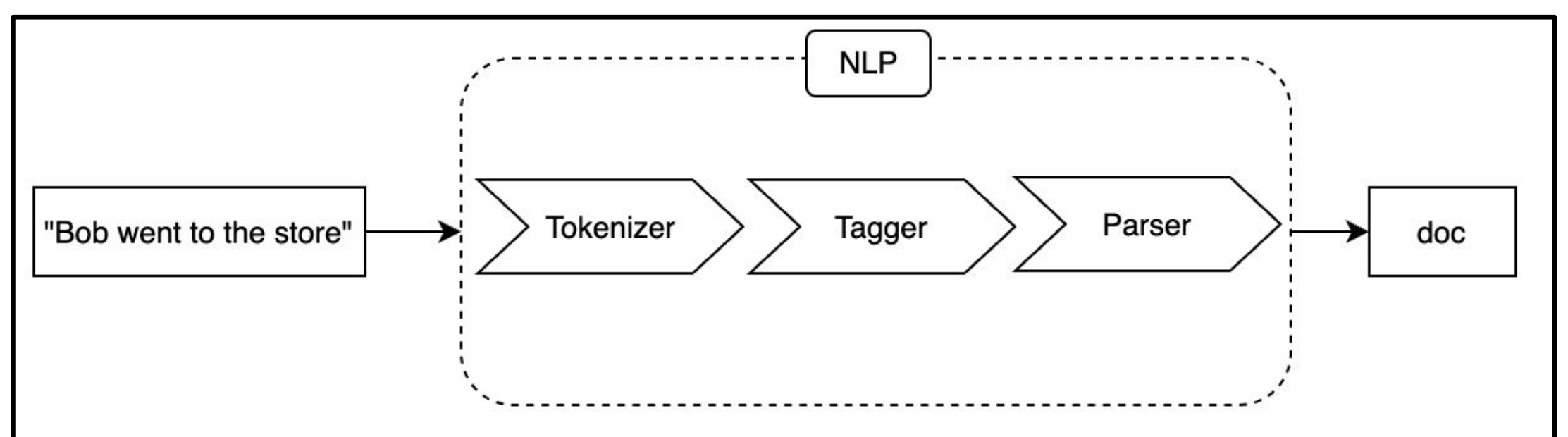
1. White-spaces (' ')
2. Most symbols (@, *, %, $, &, etc.)
3. Some number formats (11:45, 50.45, 1/2, etc.)

- About page provide important information regarding:
  - Error parameters
  - Algorithm details
  - Acknowledgment of anomalies
  - Foundational functionality

Overview of Tale Scanner Analysis (NLP)

spaCy

- Open-source library for natural language processing (NLP) in Python
- Features include tokenization, parts of speech tagging and dependency parsing. Helped accomplish analysis for clauses and morphemes



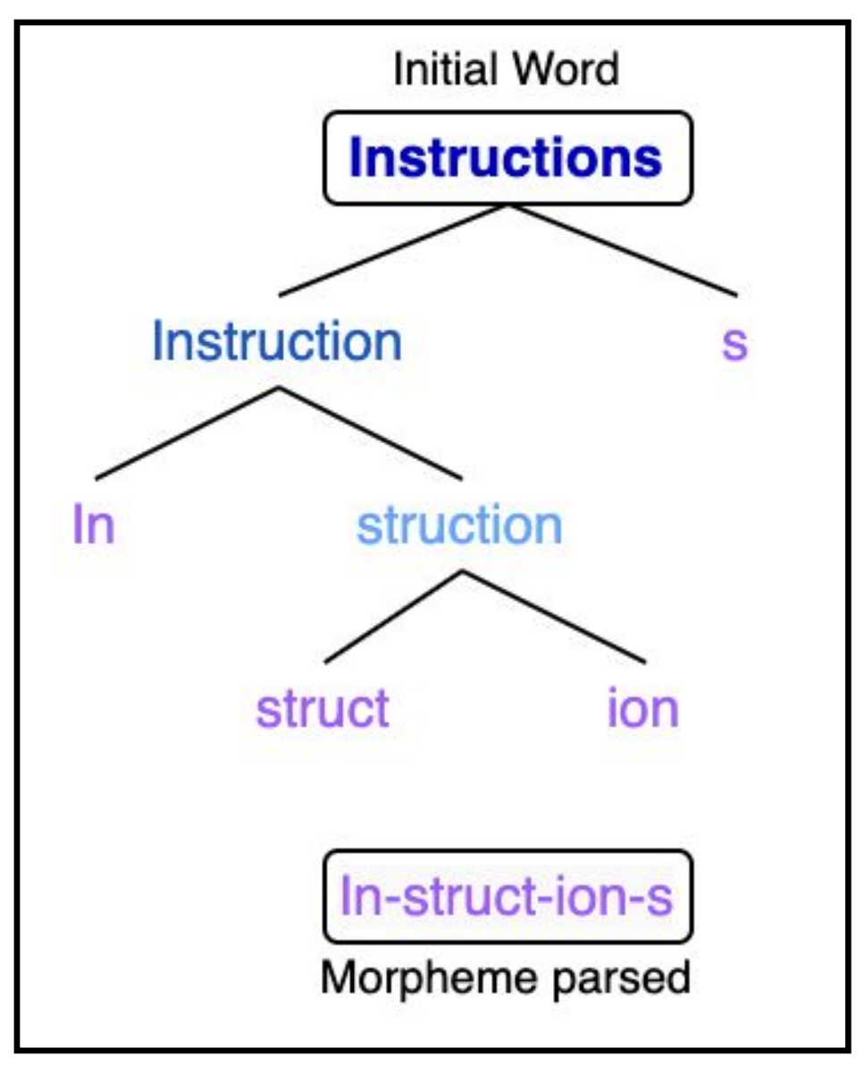
Tokenizer (token.text): [Bob, went, to, the, store]

Tagger (token.pos_): [PROPN, VERB, ADP, DET, NOUN]

Parser (token.dep_): [nsubj, ROOT, prep, det, pobj]

Morphemes

- Morphemes Python library that works in conjunction with the MorphoLex-en database
- Searches database and returns the information. Within this object is the word split up into it's morphology which includes its root, free, and bound aspects if they exist with in that word

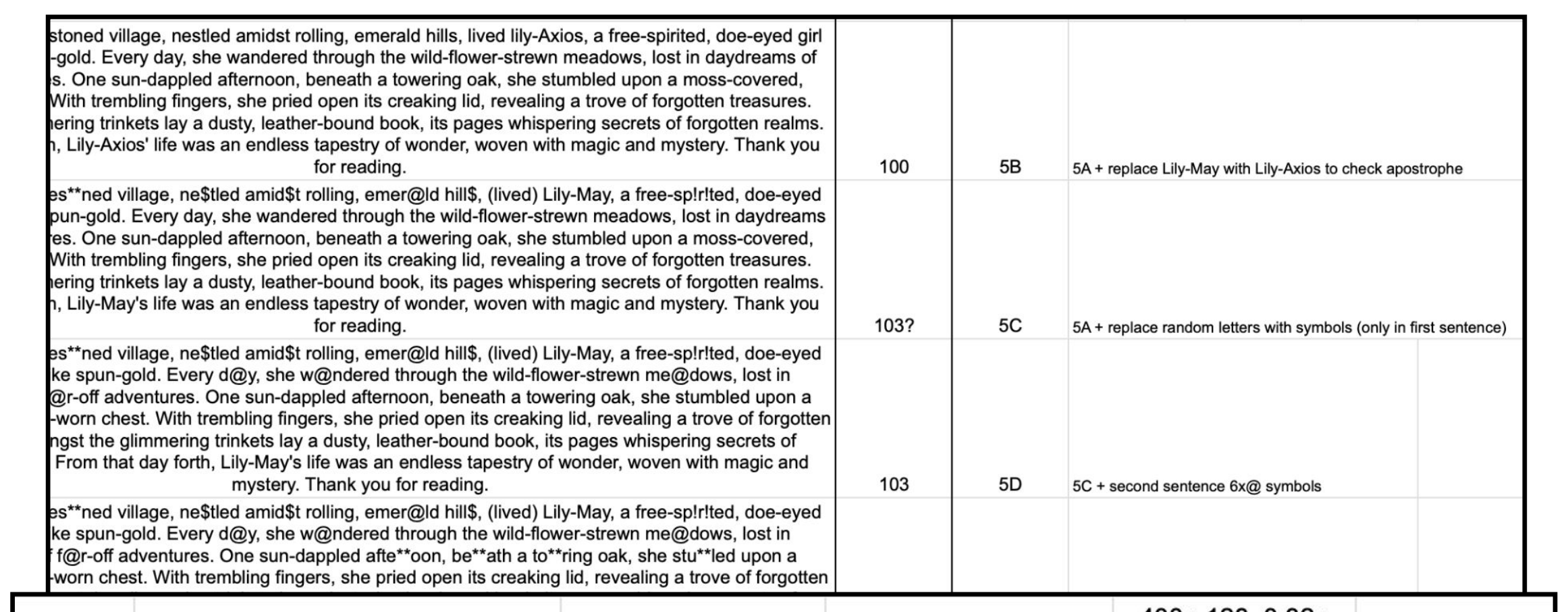


Gramformer

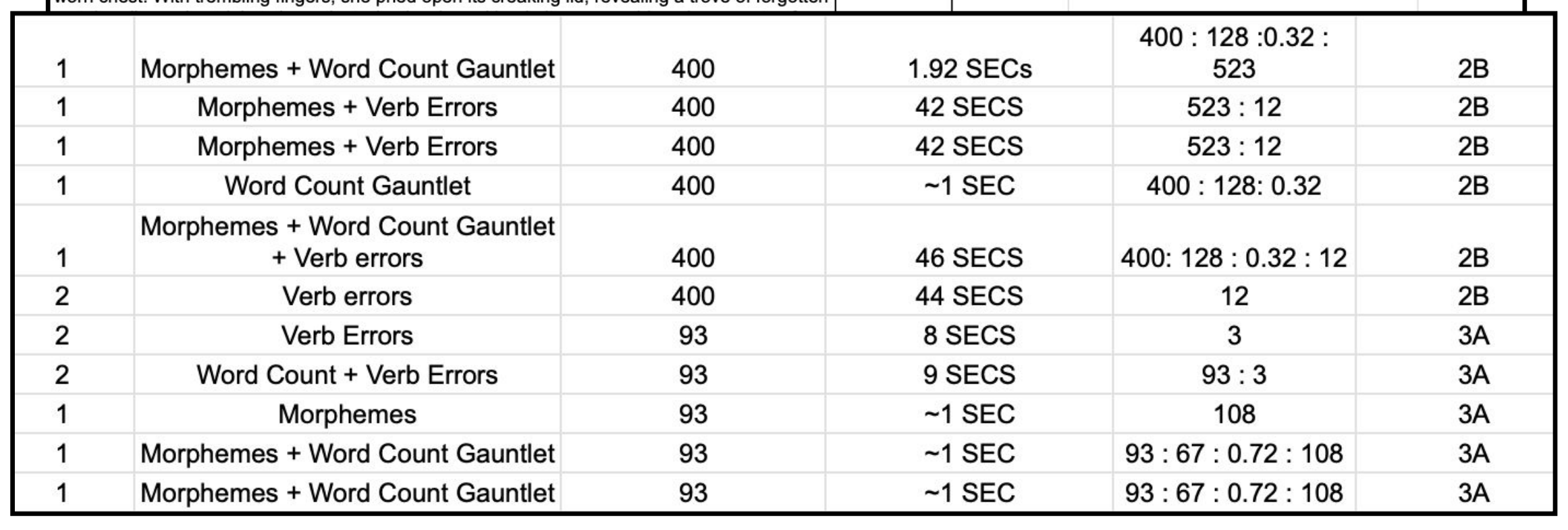
- Python library that highlights and detects grammar errors like spelling and punctuation via generative models
- Parses the user's input and then detects grammar errors, which has been learned by pre-trained and through fine tuning

Case Testing and Analysis Gaps

- Case testing was foundational to analysis optimization
- Control cases were generated to isolate and identify exception cases
- Controlled test-runs and extensive system tests were recorded to keep track of consistency
- Warnings and parameters identified through case testing



stoned village, nestled amidst rolling, emerald hills, lived lily-Axios, a free-spirited, doe-eyed girl -gold. Every day, she wandered through the wild-flower-strewn meadows, lost in daydreams of s. One sun-dappled afternoon, beneath a towering oak, she stumbled upon a moss-covered, With trembling fingers, she pried open its creaking lid, revealing a trove of forgotten treasures. ering trinkets lay a dusty, leather-bound book, its pages whispering secrets of forgotten realms. h, Lily-Axios' life was an endless tapestry of wonder, woven with magic and mystery. Thank you for reading.	100	5B	5A + replace Lily-May with Lily-Axios to check apostrophe	
es**ned village, ne$tled amid$t rolling, emer@ld hill$, (lived) Lily-May, a free-sp!r!ted, doe-eyed pun-gold. Every day, she wandered through the wild-flower-strewn meadows, lost in daydreams es. One sun-dappled afternoon, beneath a towering oak, she stumbled upon a moss-covered, With trembling fingers, she pried open its creaking lid, revealing a trove of forgotten treasures. ering trinkets lay a dusty, leather-bound book, its pages whispering secrets of forgotten realms. h, Lily-May's life was an endless tapestry of wonder, woven with magic and mystery. Thank you for reading.	103?	5C	5A + replace random letters with symbols (only in first sentence)	
es**ned village, ne$tled amid$t rolling, emer@ld hill$, (lived) Lily-May, a free-sp!r!ted, doe-eyed ke spun-gold. Every d@y, she w@ndered through the wild-flower-strewn me@dows, lost in @r-off adventures. One sun-dappled afternoon, beneath a towering oak, she stumbled upon a -worn chest. With trembling fingers, she pried open its creaking lid, revealing a trove of forgotten ngst the glimmering trinkets lay a dusty, leather-bound book, its pages whispering secrets of From that day forth, Lily-May's life was an endless tapestry of wonder, woven with magic and mystery. Thank you for reading.	103	5D	5C + second sentence 6x@ symbols	
es**ned village, ne$tled amid$t rolling, emer@ld hill$, (lived) Lily-May, a free-sp!r!ted, doe-eyed ke spun-gold. Every d@y, she w@ndered through the wild-flower-strewn me@dows, lost in f f@r-off adventures. One sun-dappled afte**oon, be**ath a to**ring oak, she stu**led upon a -worn chest. With trembling fingers, she pried open its creaking lid, revealing a trove of forgotten				



1	Morphemes + Word Count Gauntlet	400	1.92 SECs	400 : 128 :0.32 : 523	2B
1	Morphemes + Verb Errors	400	42 SECS	523 : 12	2B
1	Morphemes + Verb Errors	400	42 SECS	523 : 12	2B
1	Word Count Gauntlet	400	~1 SEC	400 : 128: 0.32	2B
1	Morphemes + Word Count Gauntlet + Verb errors	400	46 SECS	400: 128 : 0.32 : 12	2B
2	Verb errors	400	44 SECS	12	2B
2	Verb Errors	93	8 SECS	3	3A
2	Word Count + Verb Errors	93	9 SECS	93 : 3	3A
1	Morphemes	93	~1 SEC	108	3A
1	Morphemes + Word Count Gauntlet	93	~1 SEC	93 : 67 : 0.72 : 108	3A
1	Morphemes + Word Count Gauntlet	93	~1 SEC	93 : 67 : 0.72 : 108	3A

- Values recorded per test-run:
  - Reload number
  - Analysis type
  - Expected word count, expected analysis output
  - Total run-time of test-run
  - Actual output
  - Respective label code for input text

Future Enhancements

- Train large learning models for the remaining analysis types
  - Word choice errors
  - Story grammar elements
  - Cohesion
- Optimize each analysis type to handle all exception cases
- Implement multi-language analysis for necessary functions
- Handle different file input types (pdf, wordx, etc.)

Summary

- Tale Scanner was designed with the intention of aiding speech language pathologists conduct statistical analysis more efficiently
- Ten of the original thirteen requirements were successfully implemented
- Tale Scanner provides a level of consistency in analysis that is often not represented by generalized artificial intelligence
- The system is designed and built with independent stability; changes to included libraries and technologies will not affect the build
- Case testing is foundational to optimizing natural language processing, and was fundamental to Tale Scanner's success