A Part of Speech Tagger for Software Documentation

SD May 2021 Group 35

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.

Vision

Bring the power and flexibility of natural language processing to software documentation

Wide Reaching Benefits

More data for training NL ⇔ Code Generation

Ability to infer information from documentation

Possible auto generation of documentation

Requirements

Functional

- Input: previously unseen untagged software documentation
- Output: PoS tagged software documentation
- PoS tag...
  - Text with english and code mixed
  - English descriptions of code
  - Pseudocode

Non-Functional

- Create new PoS tags to represent abstract parts of code
- Retrain the Stanford NLP on tagged software documentation
- Translate the tagged data to a format readable by the Stanford NLP
- Start building up a large corpus of PoS tagged software documentation

Engineering Constraints

- Build our solution on top of and/or utilizing the Stanford NLP Pipeline
- Build our solution within two semesters
- Low to no cost
  - Stanford NLP is publicly available
  - ISU provides resources to train machine learning models if necessary

Standards

- ISO-IEC 12207: Software Lifecycle | Longevity of Project
- ISO-IEC 9001: Quality Management | Quality of Project
- ECMA 494: JSON | Data Transfer in Pipeline

Operating Environment

- Java 15 / Java SE 16
- Python 3.8 or above

Intended Users

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

Design Approach

TreeBank / Tag Set

A combination of three different tag sets

1. Penn TreeBank for English
   - *NN = noun, VBZ = verb*
2. Code Tags
   - *<val> = value, <type> = type*
3. HTML Tags
   - *<code> = HTML code tag*

Tagged Tokens Example

The function takes an array of size n, where each element $e \in \mathbb{N}$, and outputs their sum.

Input: a = [12, 3, 7]

sumArray(a)

Output: 22

Tagged Tokens Example

The function takes an array of size n , where each element $e \in \mathbb{N}$, and outputs their sum .

Input : a = [12 , 3 , 7]

sumArray (a)

Output : 22

Tagged Tokens Example

The function takes an array of size n, where each element $e \in \mathbb{N}$, and outputs their sum.

DT NN VBZ DT NN IN NN <var>, WRB DT NN <var> SYM SYM , CC NNS PRP$ NN .

Input : a = [12 , 3 , 7]

NN : <var> <gets> <[> <value> <,> <value> <,> <value> <]>

sumArray (a)

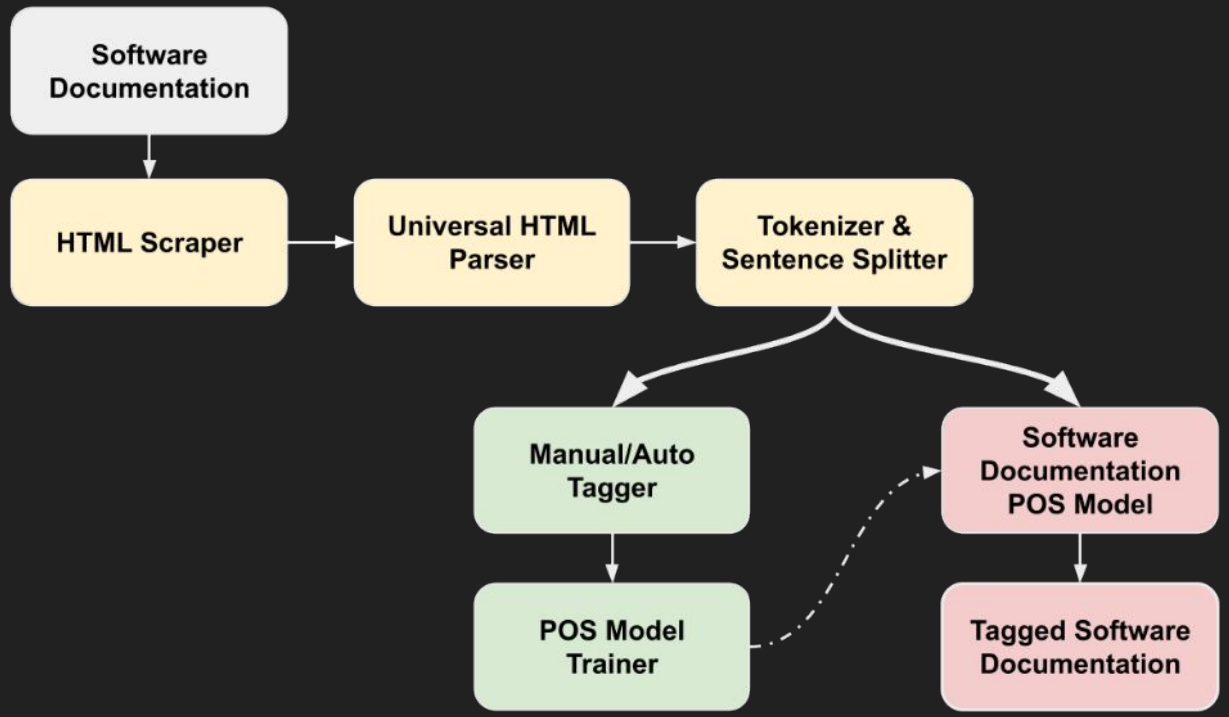
<func> <(> <param> <)>

Output : 22

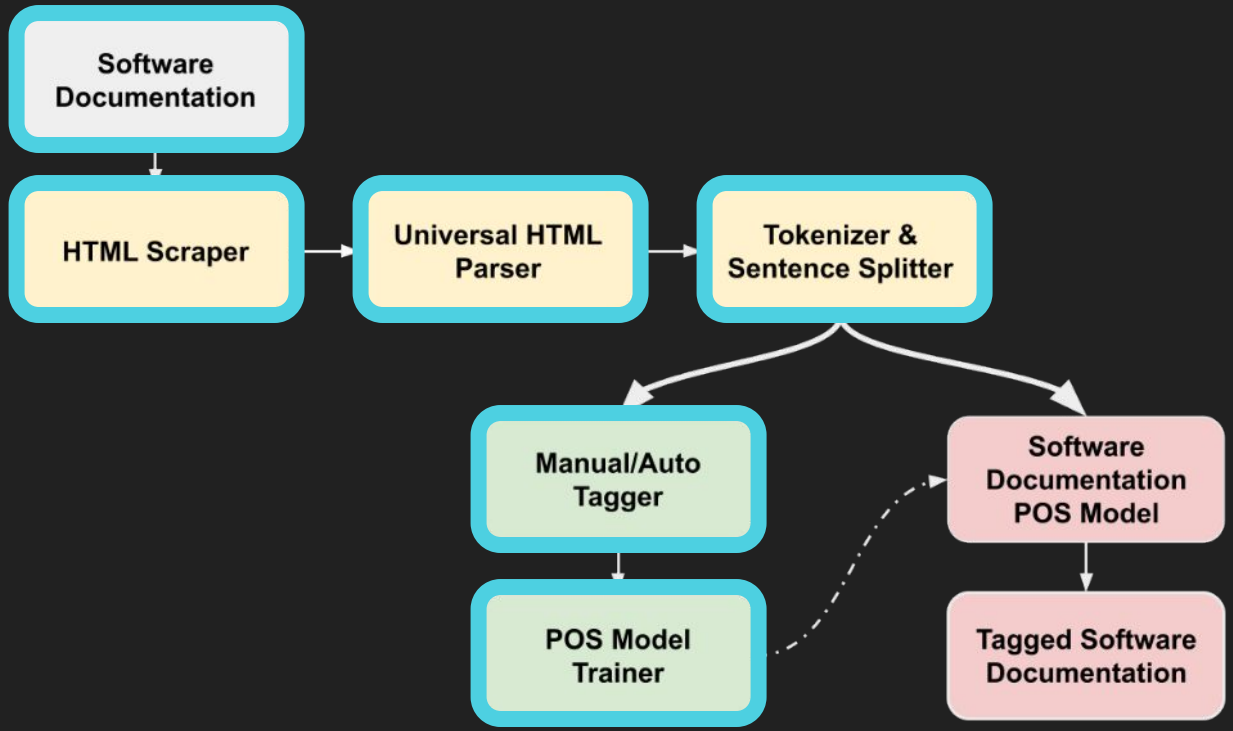
NN : <value>

**Our own tags are enclosed within angle brackets < >

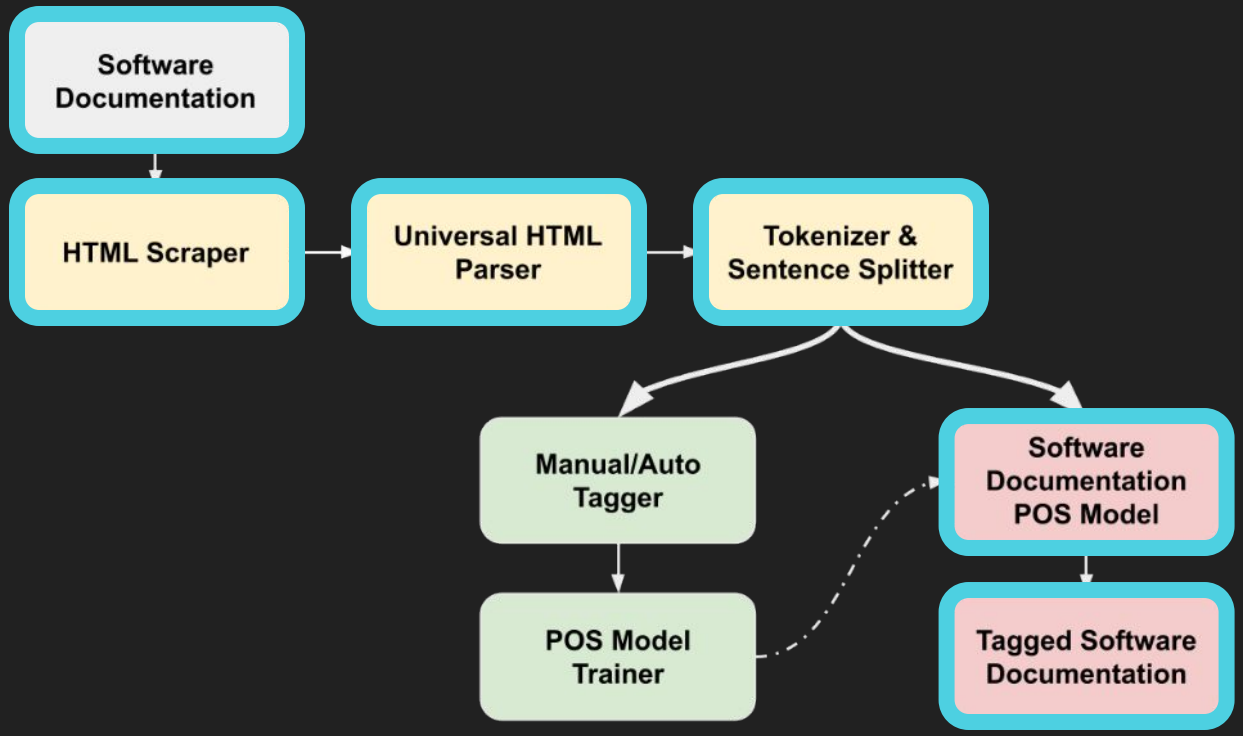
System Design - Pipeline



Training a new model



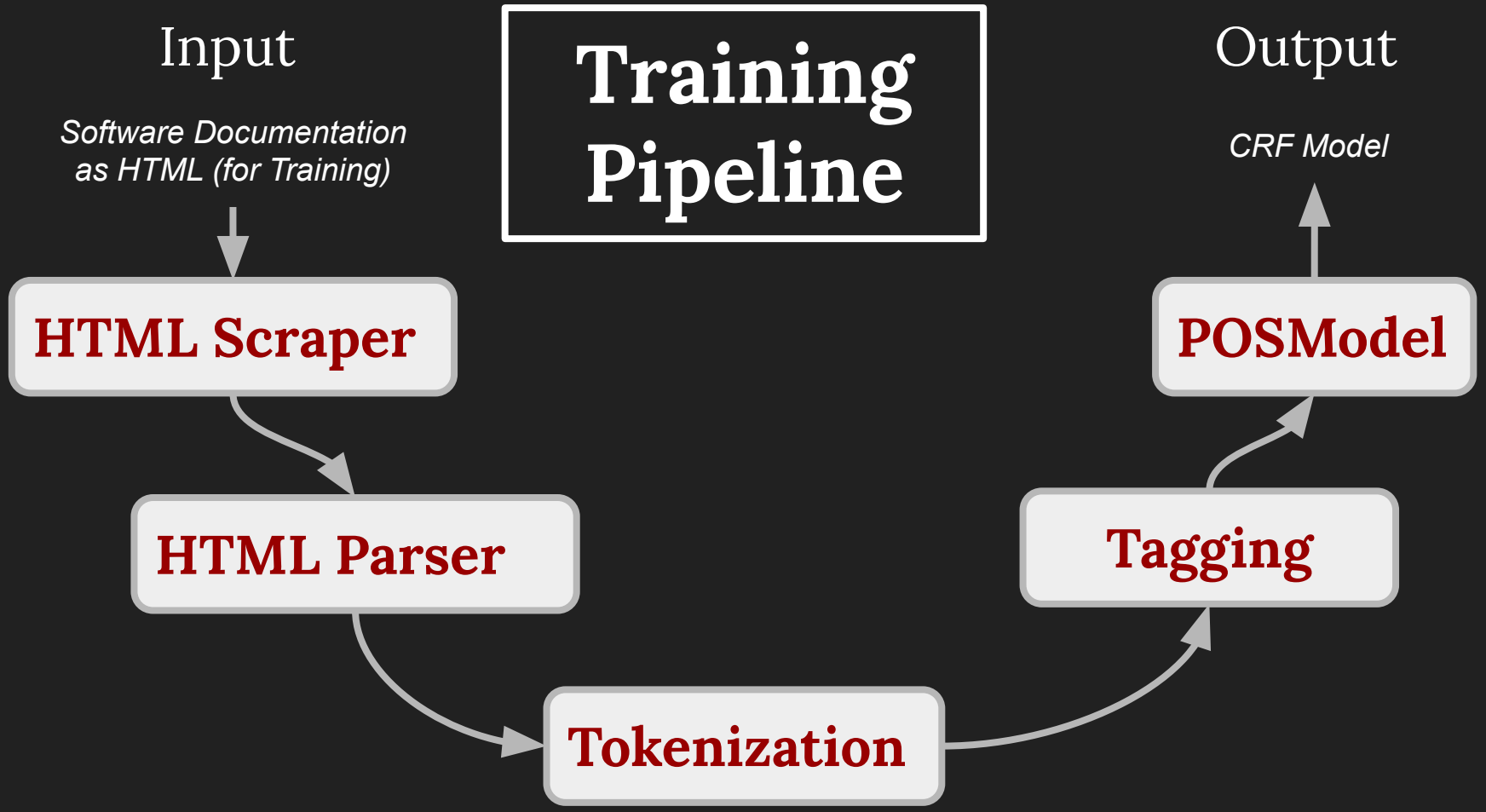
Tagging new software docs



Technical Details

and a step by step demo of our pipeline

Training Pipeline



Input

```
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Boolean.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Character.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Double.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Math.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Arrays.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Currency.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Random.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Scanner.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Stack.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Timer.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Vector.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ConcurrentModificationException.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/IllformedLocaleException.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/TooManyListenersException.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/Annotation.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/DateFormat.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/DecimalFormat.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/Format.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/NumberFormat.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/ConcurrentHashMap.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/DelayQueue.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/Exchanger.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/Semaphore.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/TimeUnit.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/ThreadLocalRandom.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/SynchronousQueue.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/ExecutionException.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/TimeoutException.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/function/BiConsumer.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/function/BinaryOperator.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/function/DoubleConsumer.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/function/Function.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/function/IntConsumer.html
```

HTML Scraper

Output

HTML Scraper

```
<!-- ======== START OF CLASS DATA ======== -->
<main role="main">
<div class="header">
<div class="subTitle"><span class="moduleLabelInType">Module</sp
<div class="subTitle"><span class="packageLabelInType">Package</
<h2 title="Interface ObjectInput" class="title">Interface Object
</div>
<div class="contentContainer">
<div class="description">
<ul class="blockList">
<li class="blockList">
<dl>
<dt>All Superinterfaces:</dt>
```

Input

```
<!-- ======== START OF CLASS DATA ======== -->
<main role="main">
<div class="header">
<div class="subTitle"><span class="moduleLabelInType">Module</sp
<div class="subTitle"><span class="packageLabelInType">Package</
<h2 title="Interface ObjectInput" class="title">Interface Object
</div>
<div class="contentContainer">
<div class="description">
<ul class="blockList">
<li class="blockList">
<dl>
<dt>All Superinterfaces:</dt>
```

HTML Parser

Parse HTML Based on a tag whitelist

Using lxml, glob

+ Tag Whitelist

HTML Parser

Parse HTML Based on a tag whitelist

Using lxml, glob

Output

```
public final class Boolean
extends Object
implements Serializable, Comparable<Boolean>
The Boolean class wraps a value of the primitive type
 <code>boolean</code> in an object. An object of type
 <code>Boolean</code> contains a single field whose type is
 <code>boolean</code>.

 In addition, this class provides many methods for
 converting a <code>boolean</code> to a <code>String</code> and a
 <code>String</code> to a <code>boolean</code>, as well as other
 constants and methods useful when dealing with a
 <code>boolean</code>.


Since:
JDK1.0
See Also:
Serialized Form
```

Input

```
public final class Boolean
extends Object
implements Serializable, Comparable<Boolean>
The Boolean class wraps a value of the primitive type
 <code>boolean</code> in an object. An object of type
 <code>Boolean</code> contains a single field whose type is
 <code>boolean</code>.

 In addition, this class provides many methods for
 converting a <code>boolean</code> to a <code>String</code> and a
 <code>String</code> to a <code>boolean</code>, as well as other
 constants and methods useful when dealing with a
 <code>boolean</code>.


Since:
JDK1.0
See Also:
Serialized Form
```

Tokenization

Tokenizes and sentence splits parsed HTML based on rules

Tokenization

Tokenizes and sentence splits parsed HTML based on rules

Output

```
2  Find all unique triplets in the array which gives the sum of zero . </p>
3  <p> Notice that the solution set must not contain duplicate triplets . </p>
```

Input


```
{
  "tokens": [
    {
      "token": "Given",
      "code": false
    },
    {
      "token": "an",
      "code": false
    },
    {
      "token": "array",
      "code": false
    },
    {
      "token": "<code>",
      "code": true
    },
    {
      "token": "nums",
      "code": true
    },
    {
      "token": "</code>",
      "code": true
    },
```


AutoTagging

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

Using stanford.nlp, javafx

ManualTagging

Tag tokens that can’t be confidently auto-tagged

AutoTagging

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

Using stanford.nlp, javafx

Output


```
"tokens": [
  {
    "token": "Given",
    "code": false,
    "tag": "VBN"
  },
  {
    "token": "an",
    "code": false,
    "tag": "DT"
  },
  {
    "token": "array",
    "code": false,
    "tag": "NN"
  },
```

Input (Training)

```
"tokens": [
  {
    "token": "Given",
    "code": false,
    "tag": "VBN"
  },
  {
    "token": "an",
    "code": false,
    "tag": "DT"
  },
  {
    "token": "array",
    "code": false,
    "tag": "NN"
  },
```

POSModel

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

Using stanford.nlp, fasterxml, apache commons & log4j

Output (Training)

POSModel

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

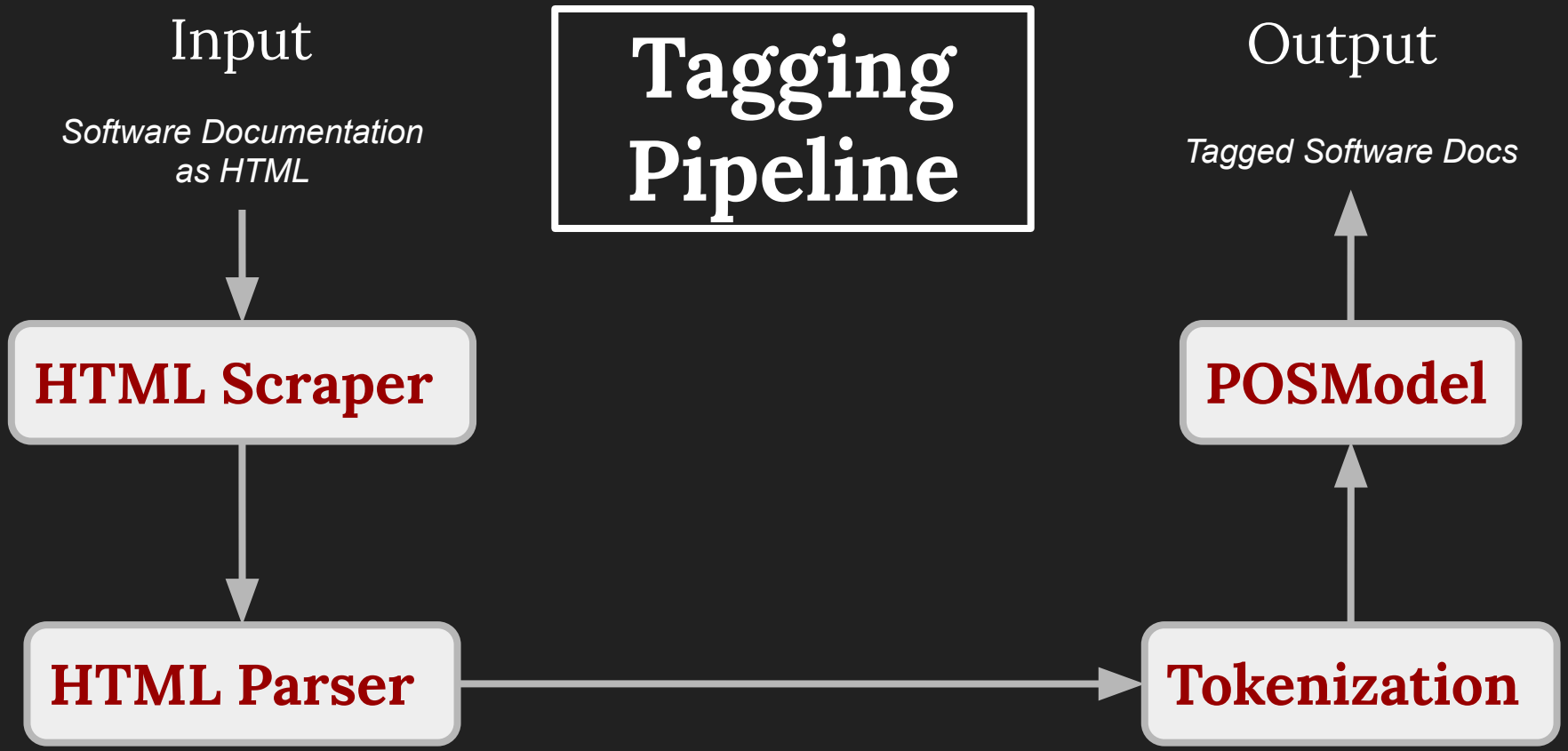
Using stanford.nlp, fasterxml, apache commons & log4j

Trained Model

Files Marked for Training

Test Results

Tagging Pipeline



Input (Tagging)

```
"tokens": [
  {
    "token": "Given"
  },
  {
    "token":"an"
  },
  {
    "token":"array"
  },
  {
    "token":"<code>"
  },
  {
    "token":"nums"
  },
  {
    "token":"</code>"
  },
  {
    "token":"of"
  }
]
```

+ **Trained Model**

POSModel

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

Using stanford.nlp, fasterxml, apache commons & log4j

POSModel

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

Using stanford.nlp, fasterxml, apache commons & log4j

Output (Tagging)


```
"tokens": [
  {
    "token": "Given"
    "tag": "VBN"
  },
  {
    "token":"an"
    "tag": "DT"
  },
  {
    "token":"array"
    "tag": "NN"
  },
  {
    "token":"nums"
    "tag":"<var>"
  },
  {
    "token":"of"
    "tag":"IN"
  }
```

Testing & Results

Integration Testing

- Individual testing for each pipeline component
  - Pipeline connections are transparent
  - Ease of testing and connectivity
- Manual Testing
  - Specific Input
  - Look for errors in execution, errors in output
- Integration Testing was superior to Unit Testing
  - Ensured pipeline communicated effectively

Model Accuracy Testing Framework

- Conversion from NLP format back into JSON format
- JSON Comparison
- Statistics and Analytics Tracking
  - Overall Accuracy
  - Missed Tokens (in order to magnitude of misses)
    - Token assignments (incorrect)
    - Individual Accuracy

Model Accuracy Testing - Results

53.61% Accurate

2606 Total Tags

Most Common Misses

- (
-)
- to
- Policy

Resolve top 4 errors:
58% Accuracy

```
[RESULTS] Accuracy: 53.61% Number of Missed Tags: 1209 Total Tags: 2606
Miss Summary
{(} missed 34 times.
    {NN=9, DT=7, JJ=1, <value>=2, IN=11, ,=1, <code>=1, .=2}
    {<(>=[<value>, <value>, ., ., IN, IN, IN, <code>, NN, NN, NN, JJ, NN], -LRB-=
{)} missed 31 times.
    {JJ=3, NN=13, DT=7, IN=4, /NN=4}
    {<)>=[DT, DT, NN, NN, NN, DT, IN, DT, IN, NN], -RRB-=[JJ, NN, IN, NN, /NN, NN
{to} missed 29 times.
    {DT=1, IN=26, ,=2}
    {TO=[DT, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, IN, I
{Policy} missed 25 times.
    {DT=15, JJ=2, NN=1, IN=4, <code>=2, /NN=1}
    {NN=[DT, DT, DT, IN, IN, IN, /NN, JJ, IN, DT, JJ, DT, DT, DT, DT, DT, DT, DT,
{.} missed 23 times.
    {<.>=1, .=22}
    {<.>=[., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., ., .], .=[<
```

Testing Conclusions

- To increase model accuracy, need more data
- Bayes based model vs CRF Classification (current)
  - Bayes based models are better with smaller data sets
  - Current model is pushing reasonable limit with data set size
- Future Expansion:
  - Significantly bigger dataset required
  - Experiment with different types of model

Faculty Advisors and Group Members

Faculty Advisors

- Ali Jannesari - Faculty Advisor
- Hung Phan - Graduate Supervisor

Group Members

- 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