Mining and Evaluating Verb tags and Other Important POS tags inside Software Documentation

William Sengstock (Team Leader), Kelly Jacobson, Zachary Witte, Jacob Kinser, Samuel Moore, Dan Vasudevan, Austin Buller

Clients: Hiep Vo, Hung Phan

Advisor: Ali Jannesari

Project Vision

- Applying Part-Of-Speech (POS) tagging to Software Documentation
- Research how to use NLP (Natural Language Processing) for software documentation POS tagging
- Common uses of POS tagging
  - Sentiment analysis - positive/negative reviews
  - Filtering spam messages
  - Language translation

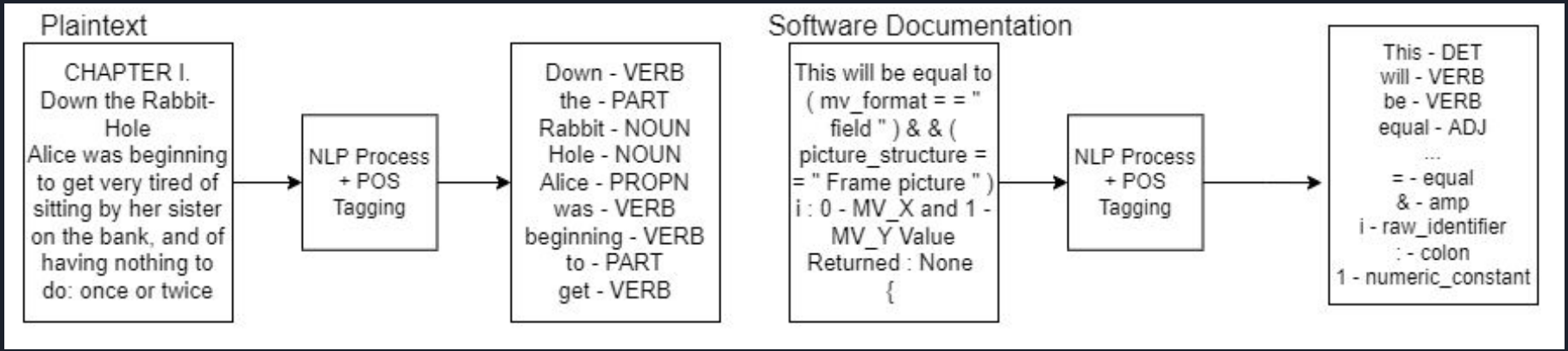
Project Vision

- How is plaintext different from software documentation?
  - Mix of plain English text and code chunks
  - Code does not follow a grammatical pattern
  - Syntax is different - brackets instead of periods
- Why use POS tags on software documentation?
  - Organization - group similar documents
  - Search - find related documents
  - Identification - what type of document are you looking at

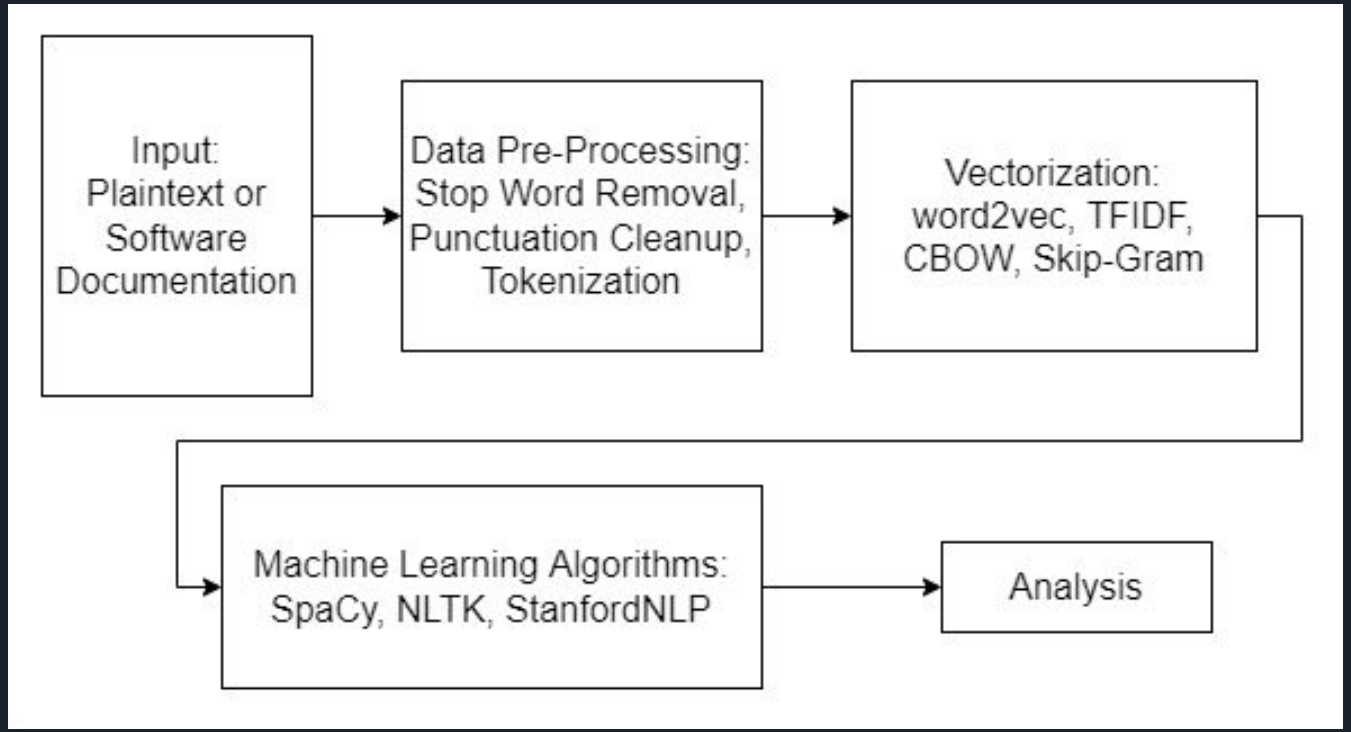
Project Requirements

- Functional
  - Given software documentation, a model can be created and relationships can be given to the data
- Non-Functional
  - The final model will have an overall tag accuracy of 90% on software documentation data
- Technical
  - A new or modified tagging system will be used to handle the software documentation

Plaintext vs Software Documentation



Conceptual Design Diagram



System Design - Preprocessing

- Tokenization
  - Breaks sentences into words
  - “This is tokenization” → [“This”, “is”, “tokenization”]
- Stop word removal
  - Ex. “the”, “a”, “an”
- Stemming
  - Removes last few characters of a word
  - Less accurate, and less costly
  - [“changing”, “changed”, “change”] → [“chang”, “chang”, “chang”]
- Lemmatization
  - Considers the context of the word
  - More accurate, but more costly
  - [“changing”, “changed”, “change”] → [“change”, “change”, “change”]

System Design - Punctuation

“A cryptocurrency, or crypto is a collection of binary data which is designed to work as a medium of exchange wherein individual coin ownership records are stored in a ledger which is a computerized database using strong cryptography to secure transaction records, to control the creation of additional coins, and to verify the transfer of coin ownership."

https://en.wikipedia.org/wiki/Cryptocurrency

In [20]:
```
import nltk
```

In [21]:
```
sample_text = "A cryptocurrency, or crypto is a collection of binary data which is designed to [] work as a medium of exchang
```

In [22]:
```
punctuation = "!()-[]{};:'\"\,<>./?@#$%^&*_~"
for item in sample_text:
    if item in punctuation:
        sample_text = sample_text.replace(item, "")
```

In [23]:
```
print(sample_text)
```

```
A cryptocurrency or crypto is a collection of binary data which is designed to  work as a medium of exchange wherein individ
ual coin ownership records are stored in a ledger which is a computerized database using strong cryptography to secure trans
action records to control the creation of additional coins and to verify the transfer of coin ownership
```

System Design - Tokenize

In [24]:
```
from nltk.tokenize import word_tokenize
tokenize = word_tokenize(sample_text)
```

In [25]:
```
print(tokenize)
```

```
['A', 'cryptocurrency', 'or', 'crypto', 'is', 'a', 'collection', 'of', 'binary', 'data', 'which', 'is', 'designed', 'to', 'w
ork', 'as', 'a', 'medium', 'of', 'exchange', 'wherein', 'individual', 'coin', 'ownership', 'records', 'are', 'stored', 'in',
'a', 'ledger', 'which', 'is', 'a', 'computerized', 'database', 'using', 'strong', 'cryptography', 'to', 'secure', 'transacti
on', 'records', 'to', 'control', 'the', 'creation', 'of', 'additional', 'coins', 'and', 'to', 'verify', 'the', 'transfer',
'of', 'coin', 'ownership']
```

In [ ]:

System Design - Stopwords

In [28]:
```
from nltk.corpus import stopwords
stopword = stopwords.words('english')
print(stopword)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'its
elf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doin
g', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'o
ut', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'has
n', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'sha
n', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [31]:
```
def remove_stopwords(tokenize):
    sample_text = [word for word in tokenize if word not in stopword]
    return sample_text
```

In [33]:
```
cleaned_data = remove_stopwords(tokenize)
print(cleaned_data)
```

```
['A', 'cryptocurrency', 'crypto', 'collection', 'binary', 'data', 'designed', 'work', 'medium', 'exchange', 'wherein', 'indi
vidual', 'coin', 'ownership', 'records', 'stored', 'ledger', 'computerized', 'database', 'using', 'strong', 'cryptography',
'secure', 'transaction', 'records', 'control', 'creation', 'additional', 'coins', 'verify', 'transfer', 'coin', 'ownership']
```

System Design - POS Tagging

In [42]:

```
for i in cleaned_data:
    wordsList = nltk.word_tokenize(i)

    tagged_data = nltk.pos_tag(wordsList)

    print(tagged_data)
```

```
[('A', 'DT')]
[('cryptocurrency', 'NN')]
[('crypto', 'NN')]
[('collection', 'NN')]
[('binary', 'NN')]
[('data', 'NNS')]
[('designed', 'VBN')]
[('work', 'NN')]
[('medium', 'NN')]
[('exchange', 'NN')]
[('wherein', 'NN')]
[('individual', 'JJ')]
[('coin', 'NN')]
[('ownership', 'NN')]
[('records', 'NNS')]
[('stored', 'VBN')]
[('ledger', 'NN')]
[('computerized', 'VBN')]
[('database', 'NN')]
[('using', 'VBG')]
[('strong', 'JJ')]
[('cryptography', 'NN')]
[('secure', 'NN')]
[('transaction', 'NN')]
[('records', 'NNS')]
[('control', 'NN')]
[('creation', 'NN')]
[('additional', 'JJ')]
[('coins', 'NNS')]
[('verify', 'NN')]
[('transfer', 'NN')]
[('coin', 'NN')]
[('ownership', 'NN')]
```

System Design - spaCy

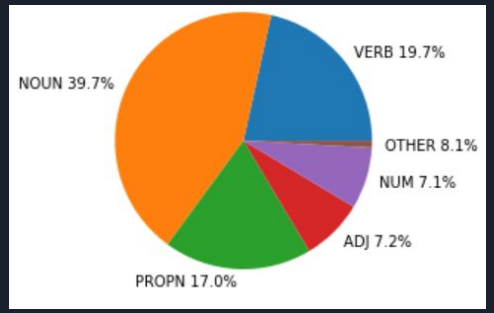
```
"number": [
    {
        "Input": 0.989533007144928
    },
    {
        "return": 0.988825798034668
    },
    {
        "Output": 0.987869918346405
    },
    {
        "integer": 0.9870409965515137
    },
    {
        "two": 0.9860666990280151
    },
    {
        "array": 0.9859938621520996
    },
    {
        "characters": 0.9859803318977356
    },
    {
        "Constraints": 0.9854423999786377
    },
    {
        "list": 0.9842109680175781
    },
    {
        "given": 0.9838504195213318
    }
],
```

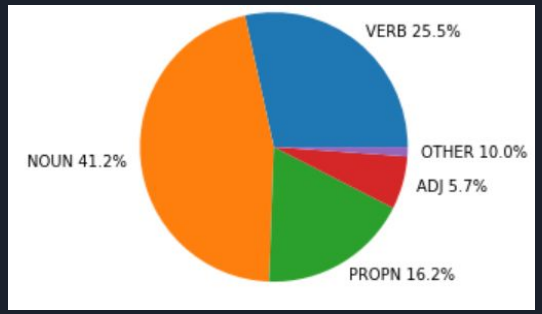
- Continuous Bag of Words (CBOW) model accuracy on software documentation
  - CBOW model uses surrounding context to predict the meaning of words

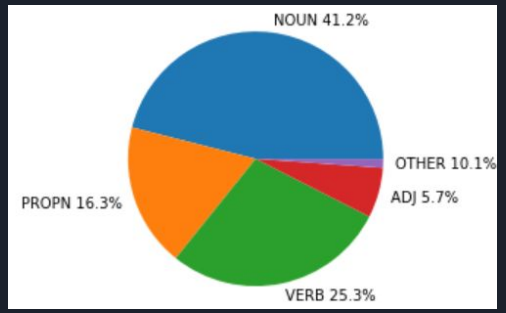
Top 10 words	No stop, no punct no num	Yes Stop Words, no num, no punct	No stop words, yes num, no punct	No stop words, yes num, yes punct	No Tokenization
Input	80%	40%	40%	20%	N/A
Example	50%	30%	40%	20%	N/A
Output	60%	90%	40%	20%	N/A
n	60%	0%	40%	20%	0%
Constraints	60%	10%	40%	20%	N/A
Explanation	50%	0%	30%	0%	N/A
return	70%	0%	30%	40%	N/A
nums	60%	30%	20%	10%	N/A
number	50%	10%	60%	30%	N/A
Given	60%	30%	40%	30%	N/A
Average	60%	24%	38%	21%	0%

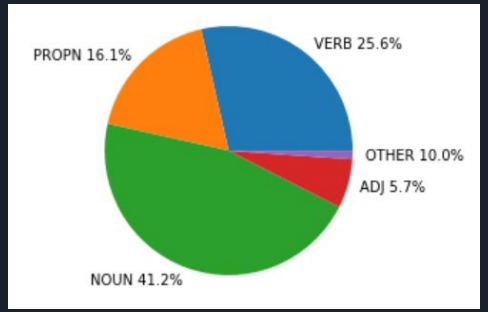
System Design - spaCy

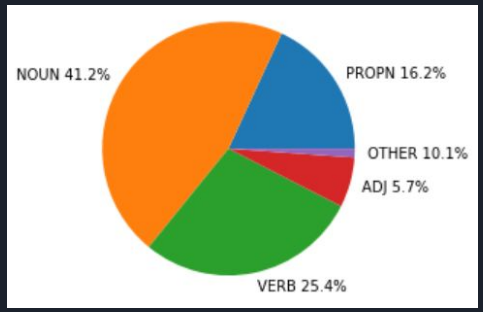
- Used KMeans clustering on software documentation
  - Test: Will spaCy cluster words with similar POS tags in the same cluster?
  - Results -





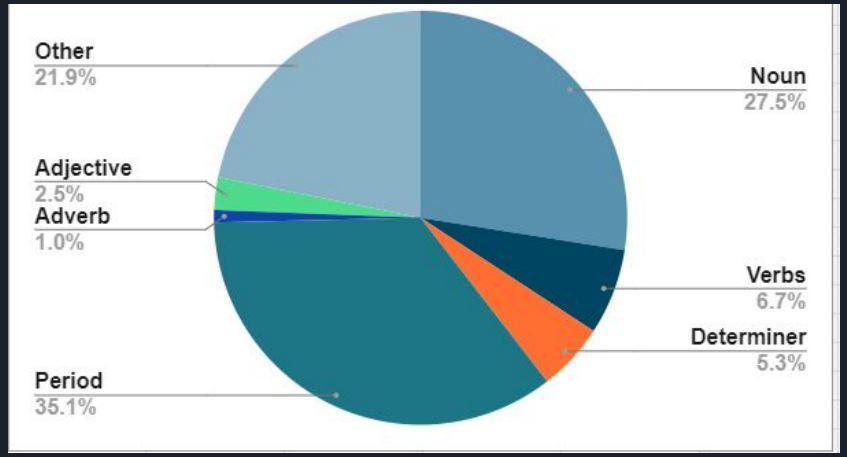


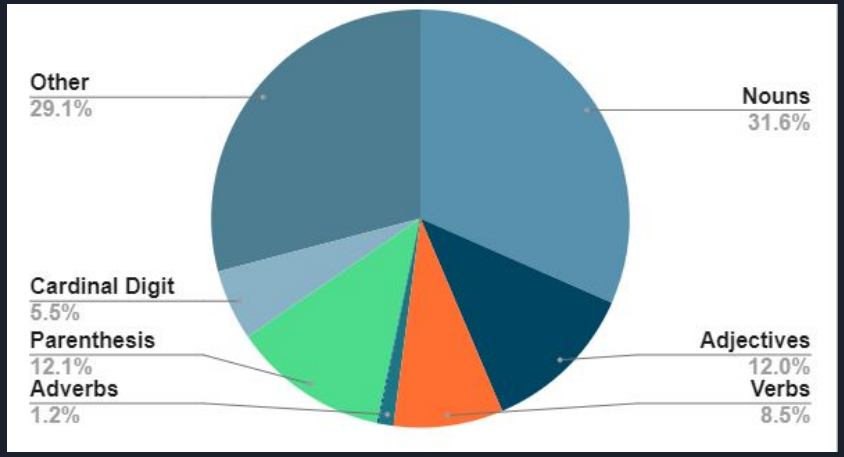




System Design - NLTK

- POSIT Data vs NLTK POS tagging accuracy
  - With Stop words removed





Design Complexity

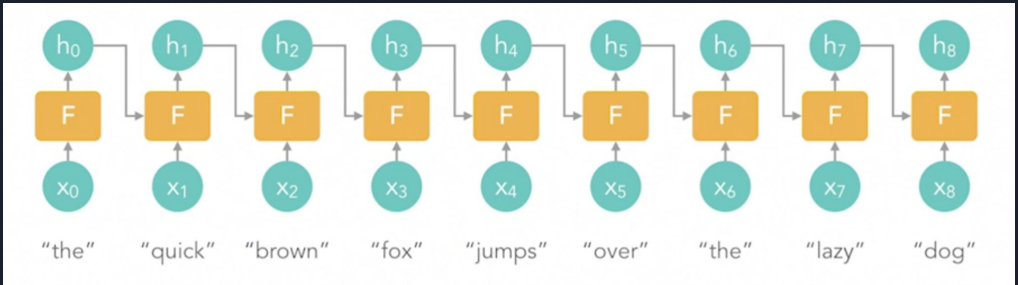
- Analyzing POS tags for software documentation
  - Sometimes tagged incorrectly
    - (Ex.) “]” tagged as “=” -> should be consistent
  - How should software documentation be tagged
- Continuous testing needed to ensure results are valid
  - POS tags are consistent with the tools used in order to determine which is more accurate.

Project Plan

- Task Decomposition
  - Part 1: Research
    - NLP basics, pre-processing data, vectorization strategies, word clustering.
    - Learning how they operate.
  - Part 2: Model Building
    - Taking research and applying them in real-time code.
    - POS tagging with different tools.
      - NLTK, SpaCY.
- Task Decomposition cont.
  - Part 3: Model Testing & Evaluation
    - Comparison of different NLP models.
    - Accuracy checks, analysis of POS tags on data, manual review.
- Risk & Mitigation Plan
  - Ensure code operates correctly
  - Apply same processing techniques when comparing models.

Alternative Approach

- Neural Networks
  - Recurrent Neural Networks (RNN)
    - Can be easily trained on paired data (POS tag, word)
    - Not many examples with software text
    - Extremely large datasets
    - Very strong ability to detect unusual patterns



Conclusion

- Progress:
  - Analysis of POS tagging toolkits for software documentation and now have a great understanding of how accurate they are in regards to software documentation
  - Experimentation with techniques we think can greatly improve model accuracy
- Next steps:
  - Use our research and experimentation to develop/train an accurate POS tagging system
  - Constantly cross examine the difference in accuracy between current POS tagging systems and our own