

Computational Analysis of Classical Texts

Marc E. Canby

Lati 318 — Cicero: *De Re Publica*
Rice University

1. Getting and Cleaning the Data
2. Exploratory Text Analysis
3. Keyword Extraction: Frequency Count and TextRank
4. Predicting Missing Text: LSTM Neural Networks

1. Getting and Cleaning the Data

2. Exploratory Text Analysis

3. Keyword Extraction: Frequency Count and TextRank

4. Predicting Missing Text: LSTM Neural Networks

Getting and Cleaning the Data

- Text obtained from *The Latin Library* at the sentence level:
 - ['nempe', 'ab', 'iis', 'qui', 'haec', 'disciplinis', 'informata', 'alia', 'moribus', 'confirmarunt', ',', 'sanxerunt', 'autem', 'alia', 'legibus', '.']
- Cleaned up messy elements of data:
 - Line numbers: [1,2,...,71]
 - Angle brackets: ['&', 'lt', ';', 'im>', ';', 'petu', 'liberavissent', ',', 'nec',...]
 - < should be < > should be >
 - Hyphens encoded as &#
 - English words: ['Cicero', 'The', 'Latin', 'Library', 'The', 'Classics', 'Page']

1. Getting and Cleaning the Data
2. Exploratory Text Analysis
3. Keyword Extraction: Frequency Count and TextRank
4. Predicting Missing Text: LSTM Neural Networks

Exploratory Text Analysis: Tokenization and POS Tagging

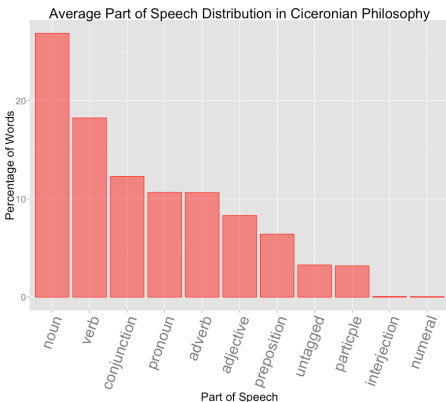
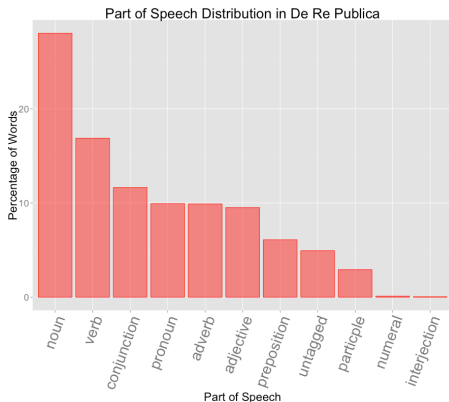
- Map each word to its base form (*lemma* or *token*) and its POS
- Often ignore *stop words* ('et', 'sum', etc.) – highlighted in red

```
['nempe', 'ab', 'iis', 'qui', 'haec', 'disciplinis',  
'informata', 'alia', 'moribus', 'confirmarunt', ',',  
'sanxerunt', 'autem', 'alia', 'legibus', '.']
```

```
[('nempe', 'adverb'), ('ab', 'preposition'), ('is',  
'pronoun'), ('qui', 'pronoun'), ('hic', 'pronoun'),  
('disciplina', 'noun'), ('informo', 'noun'), ('alius2',  
'adjective'), ('mos', 'noun'), ('confirmo', 'verb'),  
('sancio', 'verb'), ('autem', 'conjunction'), ('alius2',  
'adjective'), ('lex', 'noun')]
```

Exploratory Text Analysis

- Number of characters: 109,777 (average: 136,893)
- Number of words: 20,067 (average: 24,924)
- Number of sentences: 820 (average: 1,059)



1. Getting and Cleaning the Data
2. Exploratory Text Analysis
3. Keyword Extraction: Frequency Count and TextRank
4. Predicting Missing Text: LSTM Neural Networks

Keyword Extraction: Frequency Count

- Goal: Determine a set of keywords that summarizes the text
- Naive approach: Take words in text with highest frequency

Word	Frequency	Word	Frequency
res	191	magnus	74
publicus	163	civitas	72
Scipio	110	bonus	70
populus	103	Laelius	66
homo	94	rex	51

Note: Restricted to nouns and adjectives only

- Problem: Does not account for structure of text and relationships between words

Keyword Extraction: TextRank

TextRank is a popular keyword extraction algorithm that determines the importance of a keyword based on its relationship to other keywords.

- Step 1: Build graph representing text
- Step 2: Run keyword extraction algorithm on graph
- Step 3: Merge neighboring keywords and recompute scores

TextRank: Step 1 (Build graph representing text)

```
['nempe', 'ab', 'iis', 'qui', 'haec', 'disciplinis', 'informata', 'alia',  
'moribus', 'confirmarunt', ',', 'sanxerunt', 'autem', 'alia', 'legibus',  
'..']
```

```
[('nempe', 'adverb'), ('ab', 'preposition'), ('is', 'pronoun'), ('qui',  
'pronoun'), ('hic', 'pronoun'), ('disciplina', 'noun'), ('informo',  
'noun'), ('alius2', 'adjective'), ('mos', 'noun'), ('confirmo', 'verb'),  
('sancio', 'verb'), ('autem', 'conjunction'), ('alius2', 'adjective'),  
('lex', 'noun')]
```

```
[('disciplina', 'noun'), ('informo', 'noun'), ('alius2', 'adjective'),  
('mos', 'noun'), ('lex', 'noun')]
```

1. Getting and Cleaning the Data
2. Exploratory Text Analysis
3. Keyword Extraction: Frequency Count and TextRank
4. Predicting Missing Text: LSTM Neural Networks