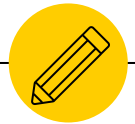
Who wrote it?

Speech Technologies Project, Spring 2016



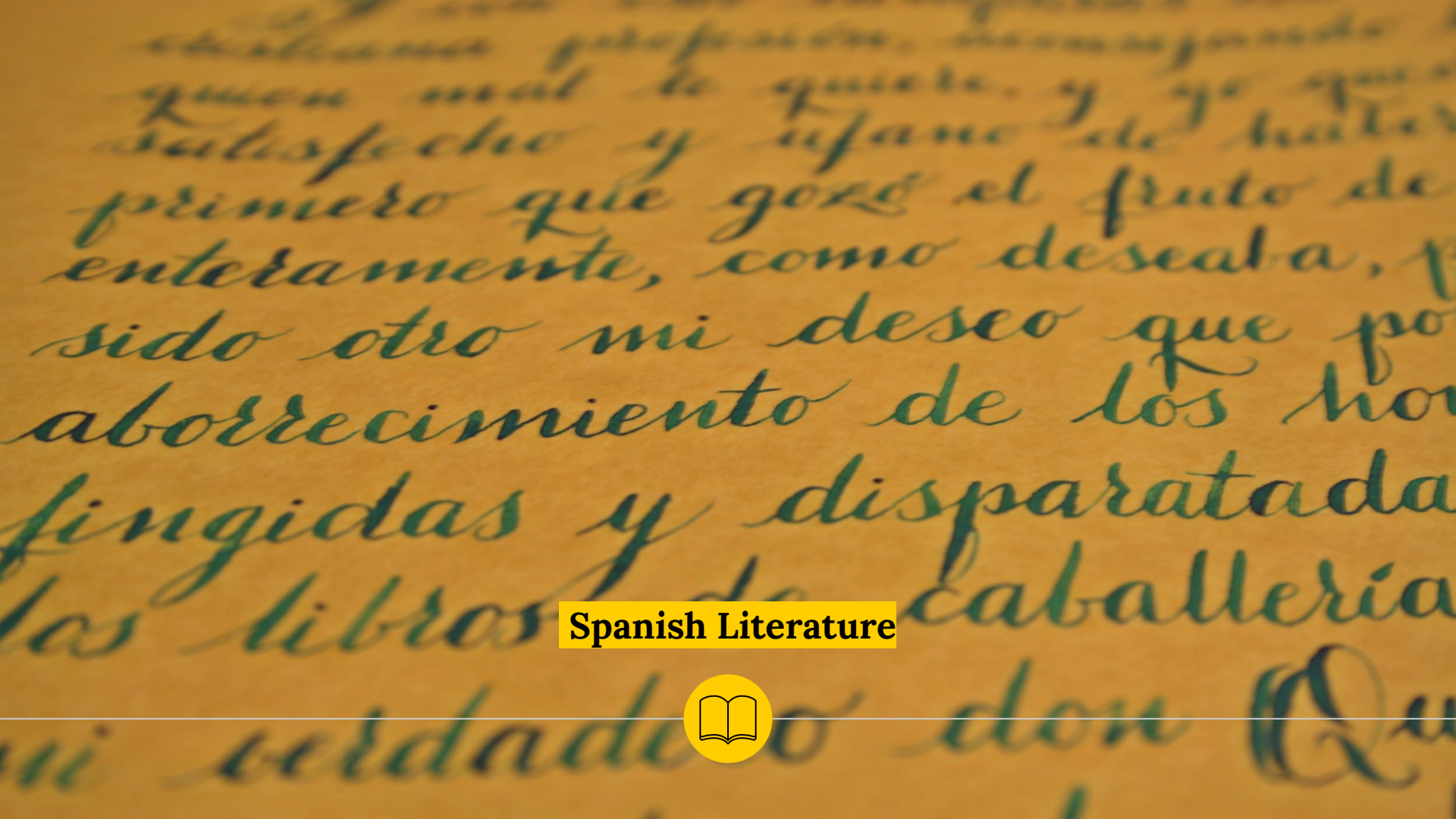
Lucas Rodés

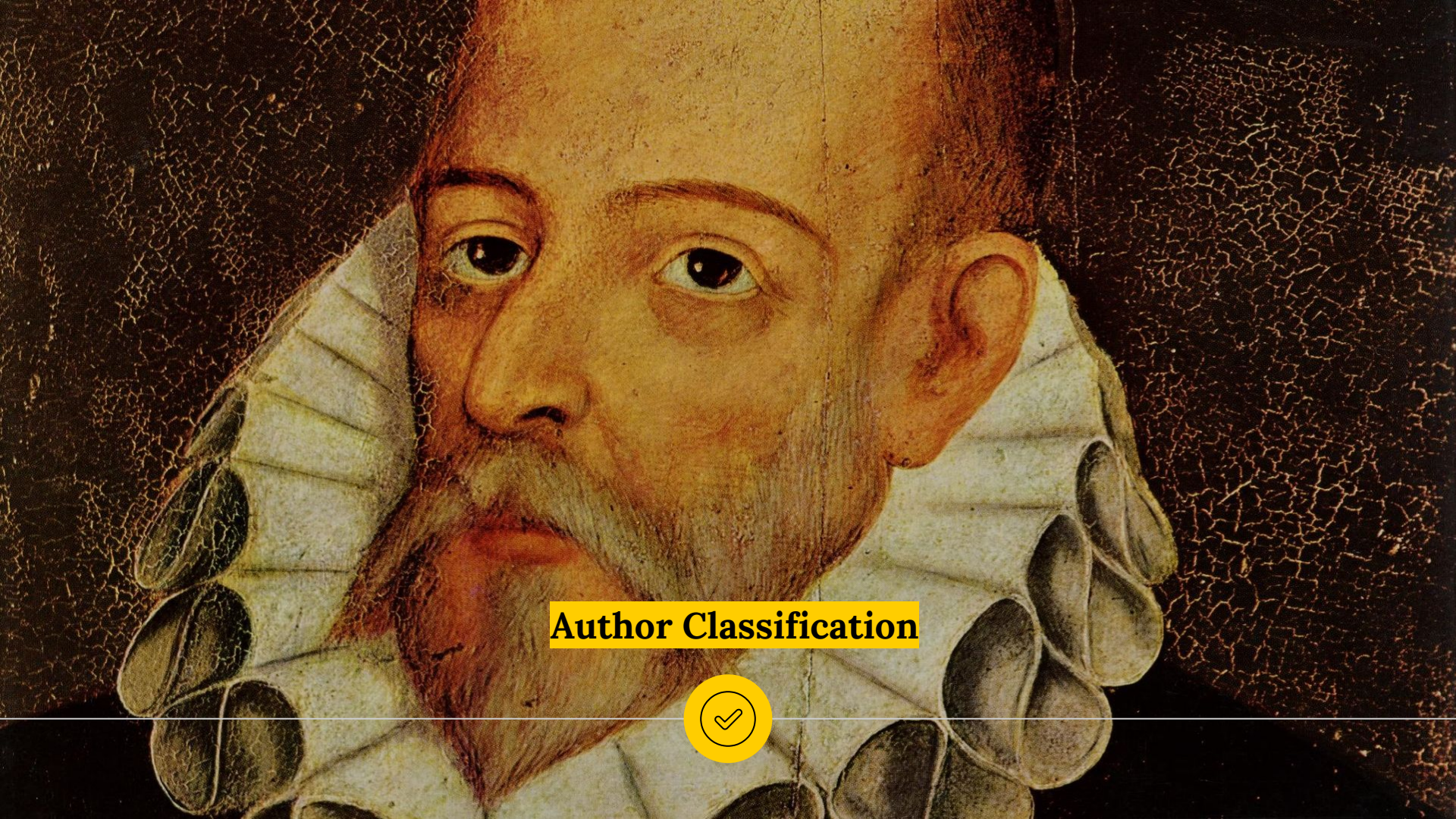
Sergi Liesegang

Outline

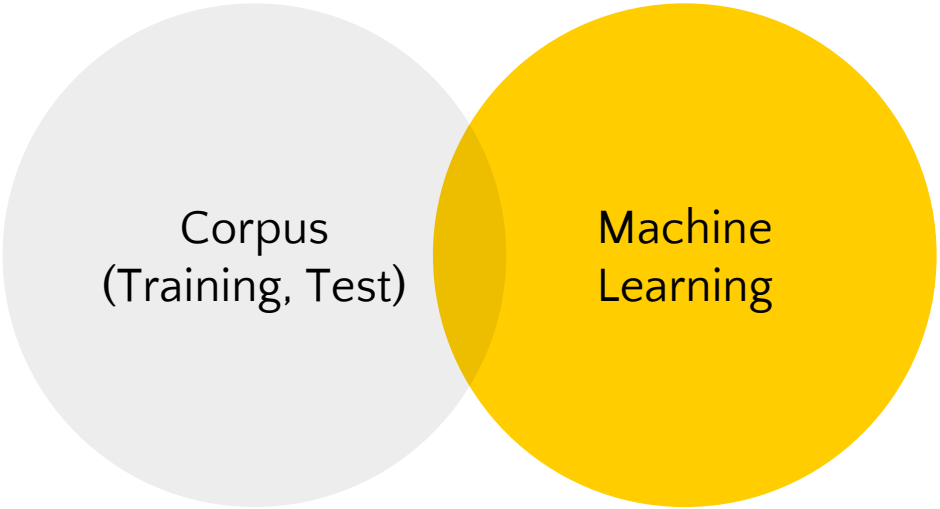
- Introduction
- Corpus
- Classifiers
- Results
- Conclusions. Future Work.

1 Introduction

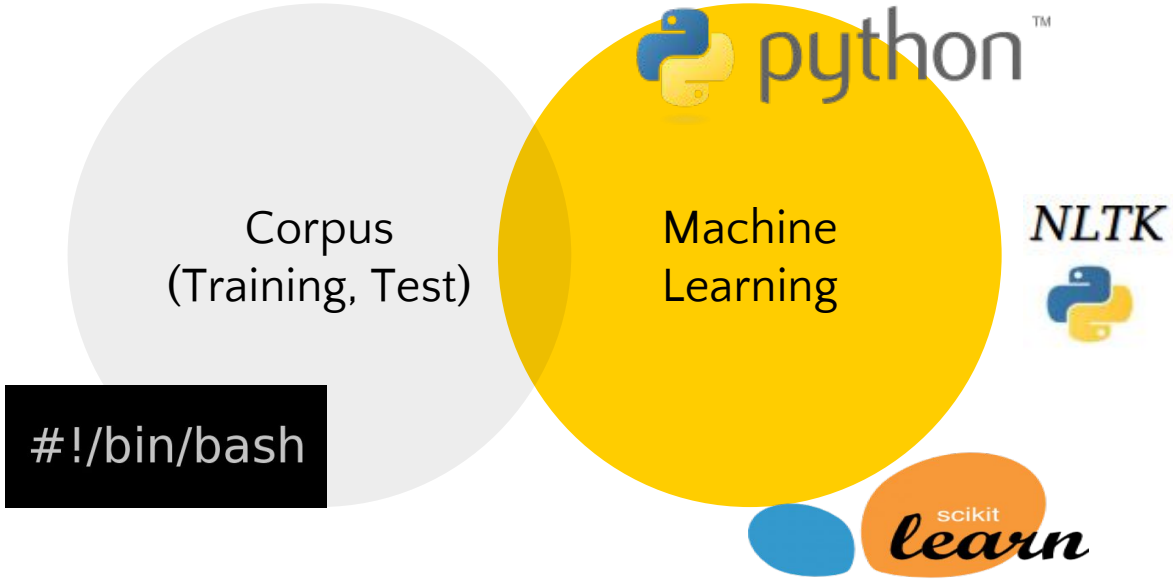

Spanish Literature


Author Classification

1 Scope



1 Tools


Corpus
(Training, Test)
Machine
Learning
python
NLTK
#!/bin/bash
scikit
learn

2 Corpus

2

Database

Author	Epoch	Genre
Miguel de **Cervantes**	Renaissance (C16-17th)	Novel - Poetry - Drama
P. **Calderón de la Barca**	Golden Age - Baroque (C17th)	Drama - Poetry
Felix **Lope de Vega**	Golden Age - Baroque (C17th)	Novel - Poetry - Drama
Francisco de **Quevedo**	Baroque (C17th)	Poetry
G. Adolfo **Becquer**	Romanticism (C19th)	Poetry - Novel
Emilia **Pardo** Bazan	Realism - Naturalism (C19th)	Novel
Benito Pérez **Galdós**	Realism (C19th)	Novel - Drama
Federico **Garcia Lorca**	Generation of 27 (C20th)	Poetry - Drama

2

Database

Author	Epoch	Genre
Miguel de **Cervantes**	Renaissance (C16-17th)	Novel - Poetry - Drama
P. Calderón de la Barca	**Golden Age - Baroque (C17th)**	**Drama - Poetry**
Felix Lope de Vega	**Golden Age - Baroque (C17th)**	**Novel - Poetry - Drama**
Francisco de **Quevedo**	Baroque (C17th)	Poetry
G. Adolfo **Becquer**	Romanticism (C19th)	Poetry - Novel
Emilia **Pardo** Bazan	Realism - Naturalism (C19th)	Novel
Benito Pérez **Galdós**	Realism (C19th)	Novel - Drama
Federico **Garcia Lorca**	Generation of 27 (C20th)	Poetry - Drama

Correlated

3 Classifiers

3

Naive Bayes

Mainstay in text Processing

Simple Categorization

Based on Bag of Words model

Compares elements in texts

3

Naive Bayes

Mainstay in text Processing

Simple Categorization

Based on Bag of Words model

Compares elements in texts

Sparse Matrix as input feature

Word occurrences

3

Naive Bayes

Mainstay in text Processing

Simple Categorization

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Based on Bag of Words model

Compares elements in texts

Sparse Matrix as input feature

Word occurrences

3 Naive Bayes

Mainstay in text Processing

Simple Categorization

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Based on Bag of Words model

Compares elements in texts

Sparse Matrix as input feature

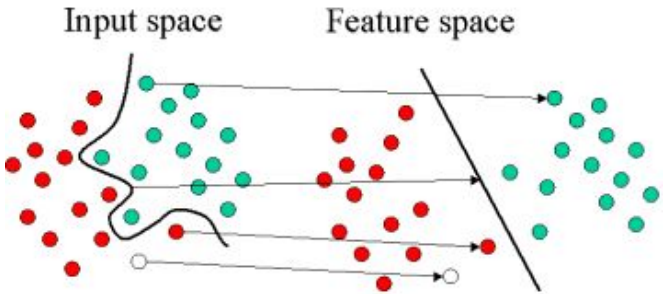
Word occurrences

Implemented in Python

Scikit-learn Library

3

Support Vector Machine



Well-known classifier

Finds boundary with maximum class separability.

Kernel

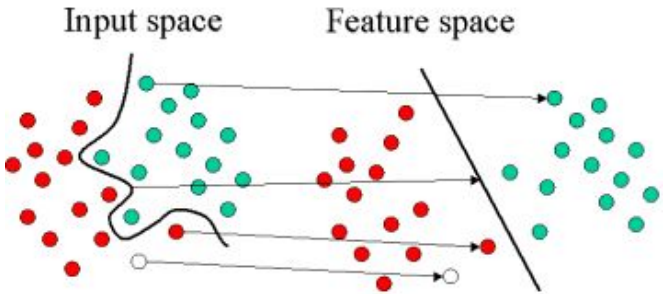
We used **Linear**

Sparse Matrix as input feature

Word occurrences

3

Support Vector Machine



Well-known classifier

Finds boundary with maximum class separability.

Kernel

We used **Linear**

Sparse Matrix as input feature

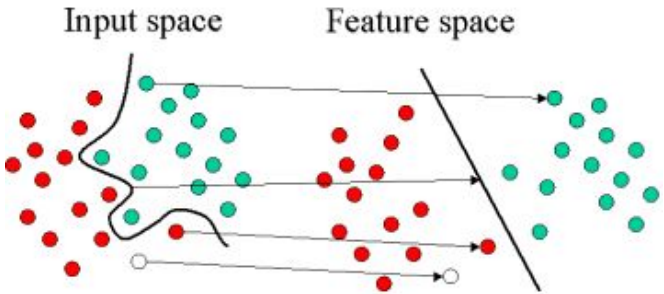
Word occurrences

Grid Search

Optimization method to find best classifier parameters

3

Support Vector Machine



Well-known classifier

Finds boundary with maximum class separability.

Kernel

We used **Linear**

Sparse Matrix as input feature

Word occurrences

Grid Search

Optimization method to find best classifier parameters

Implemented in Python

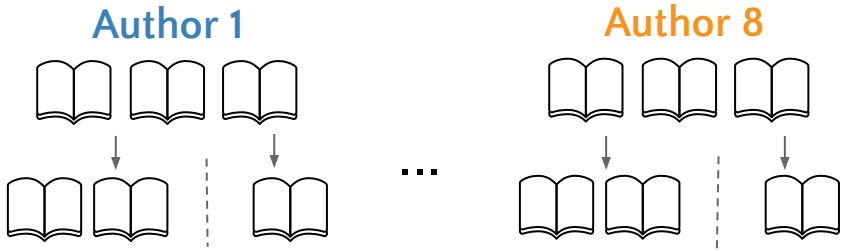
Scikit-learn Library

3 Approaches

- **Download** books

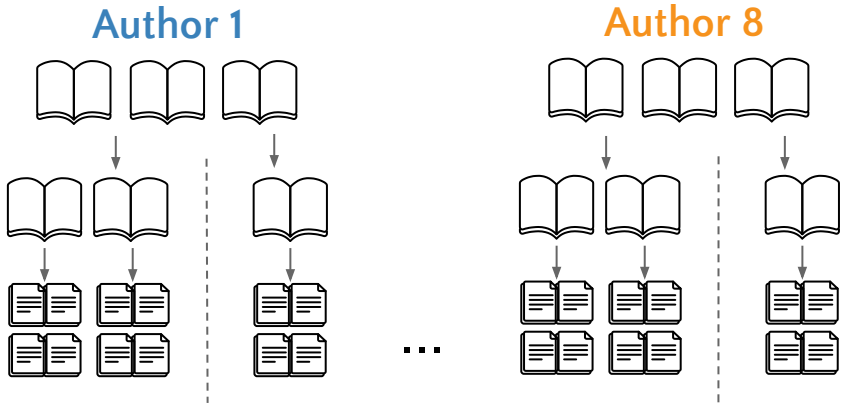
3 Approaches

- **Download** books
- 66% Training, 33% Test



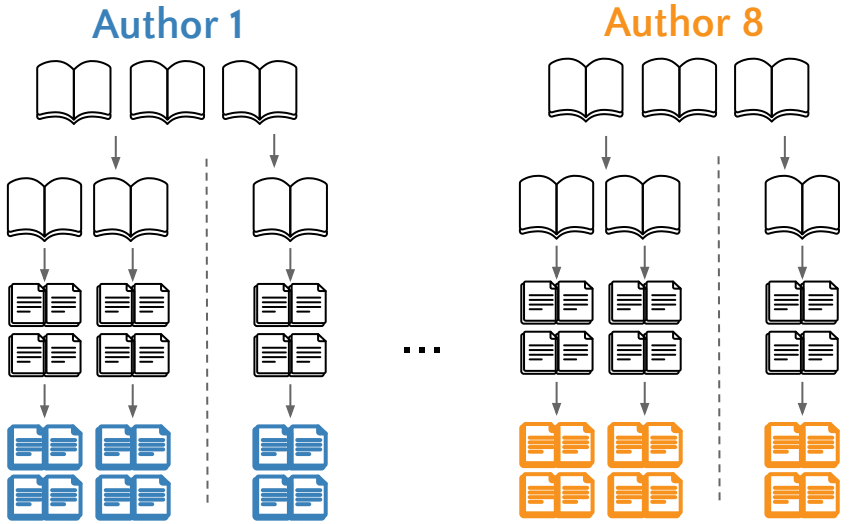
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** in chunks of *L* lines



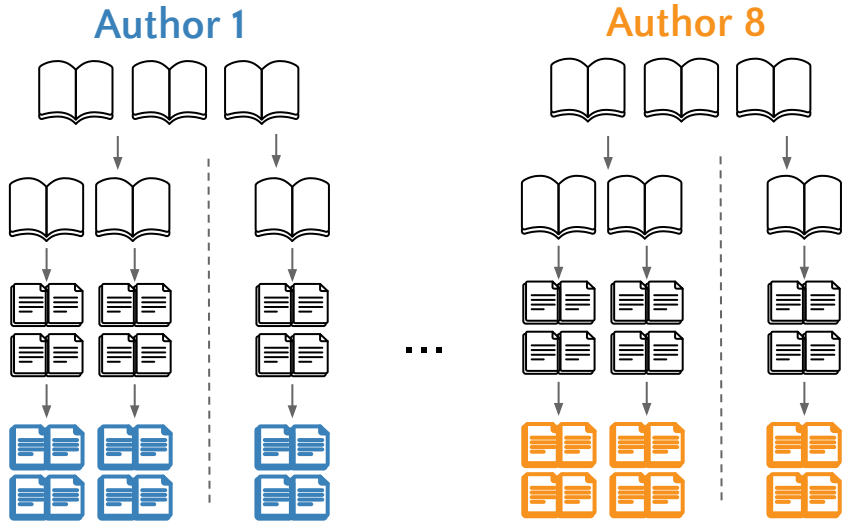
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** in chunks of *L* lines
- Labelling



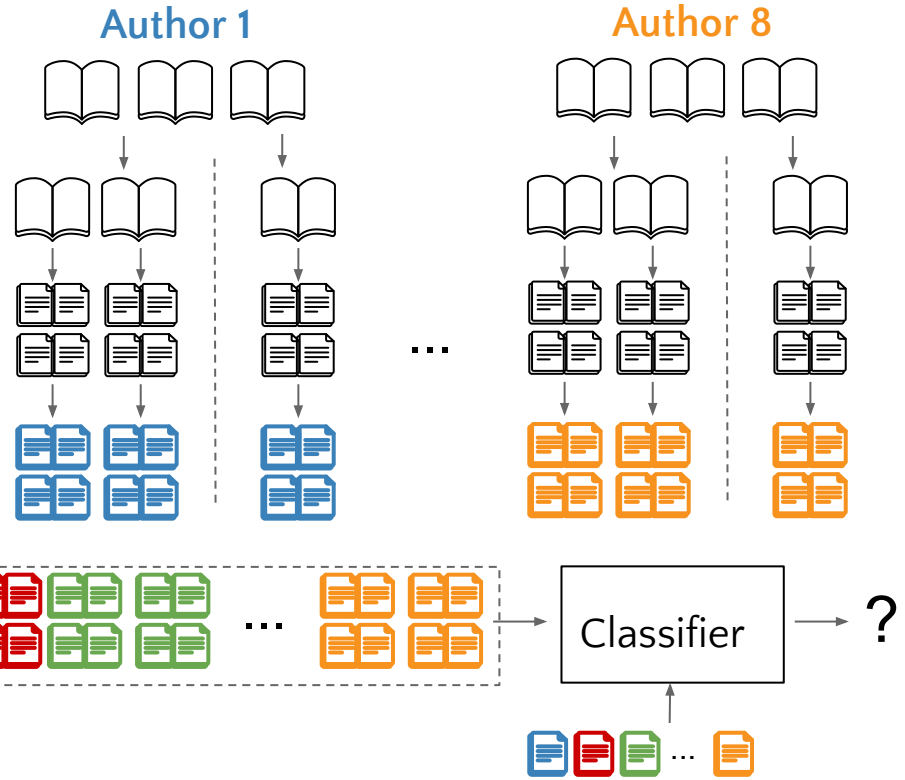
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** in chunks of *L* lines
- Labelling
- **Train Classifier**



3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** in chunks of *L* lines
- Labelling
- **Train Classifier**
- Test Classifier



3

Natural Language Modeling

Popular in speech processing

Based on Statistical Learning

Word statistics

3

Natural Language Modeling

Popular in speech processing

Based on Statistical Learning

Word statistics

N-gram Model

Unigram, Bigram,
Trigram.

3

Natural Language Modeling

Popular in speech processing

Based on Statistical Learning

Word statistics

N-gram Model

Unigram, Bigram, Trigram.

Smoothing Methods

We have used Add one Smoothing

3

Natural Language Modeling

Popular in speech processing

Based on Statistical Learning

Word statistics

N-gram Model

Unigram, Bigram, Trigram.

Smoothing Methods

We have used Add one Smoothing

Implemented in Python

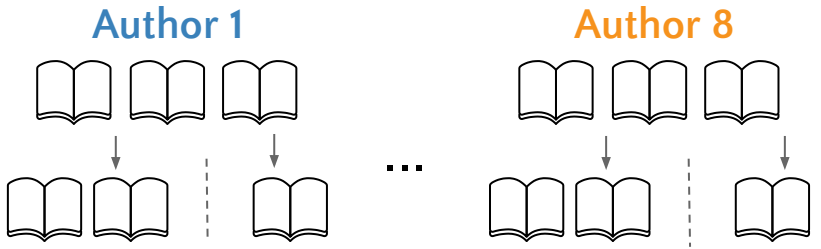
NLTK Library

3 Approaches

- **Download** books

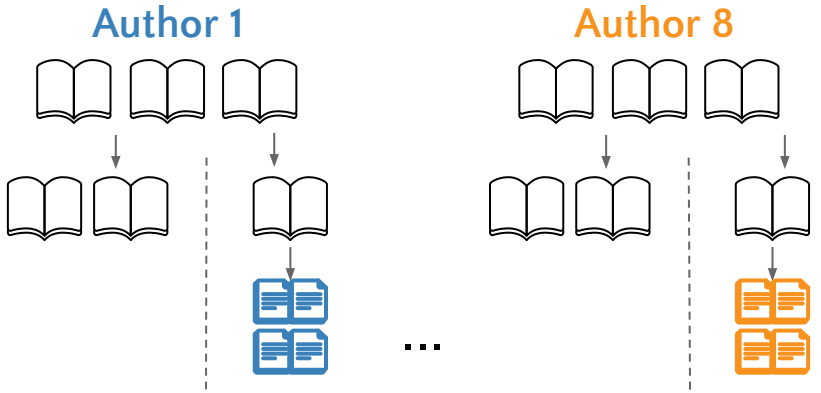
3 Approaches

- **Download** books
- 66% Training, 33% Test



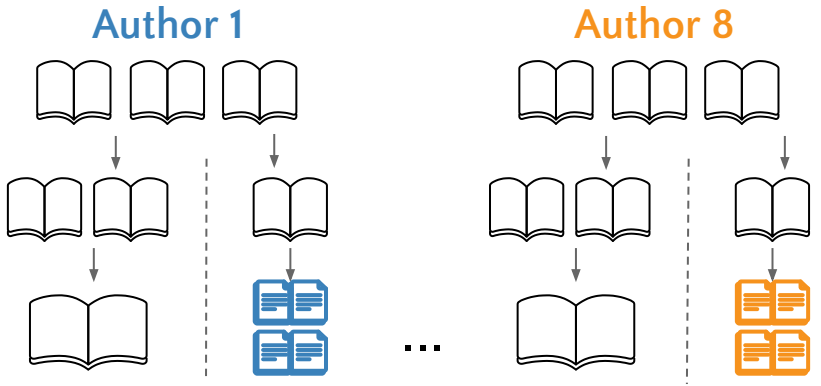
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** Test in chunks & label



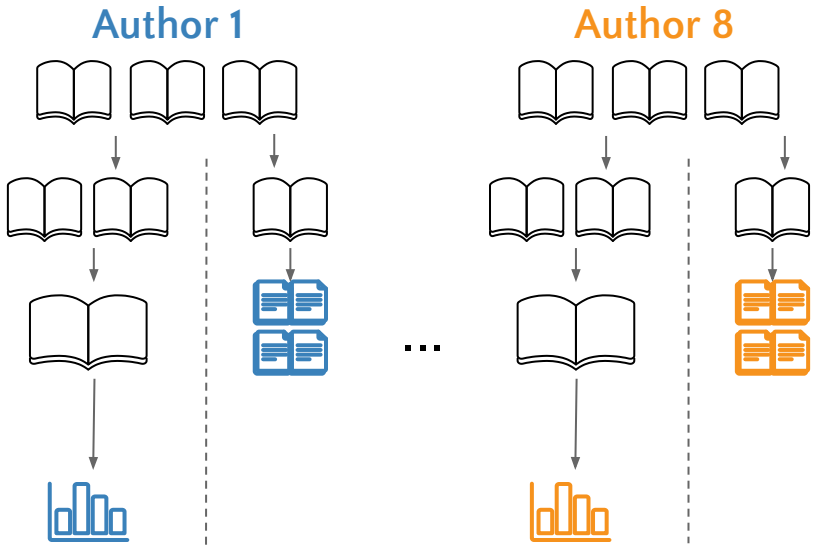
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** Test in chunks & label
- **Merge** Training books



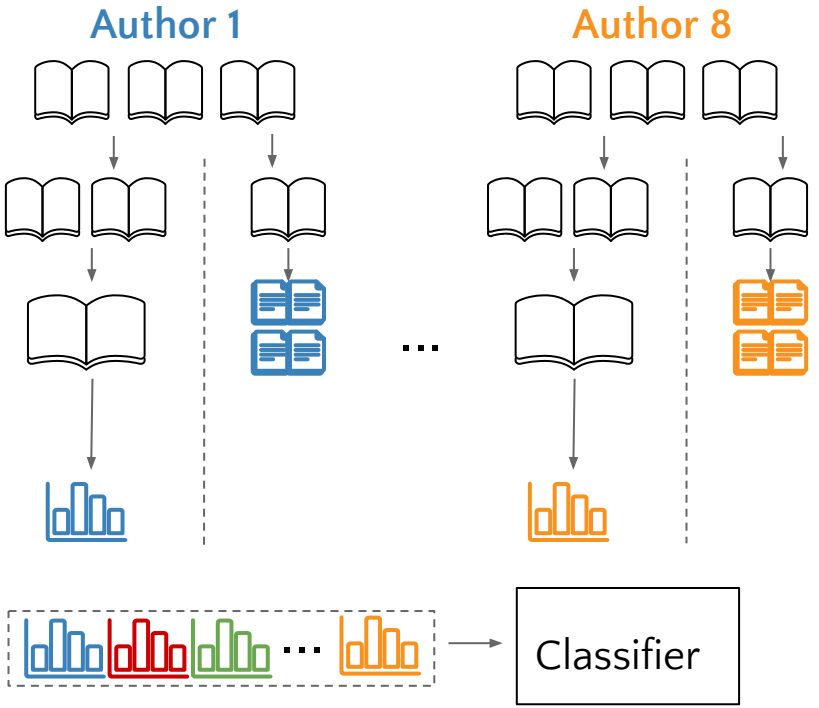
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** Test in chunks & label
- **Merge** Training books
- Obtain **Language Model**



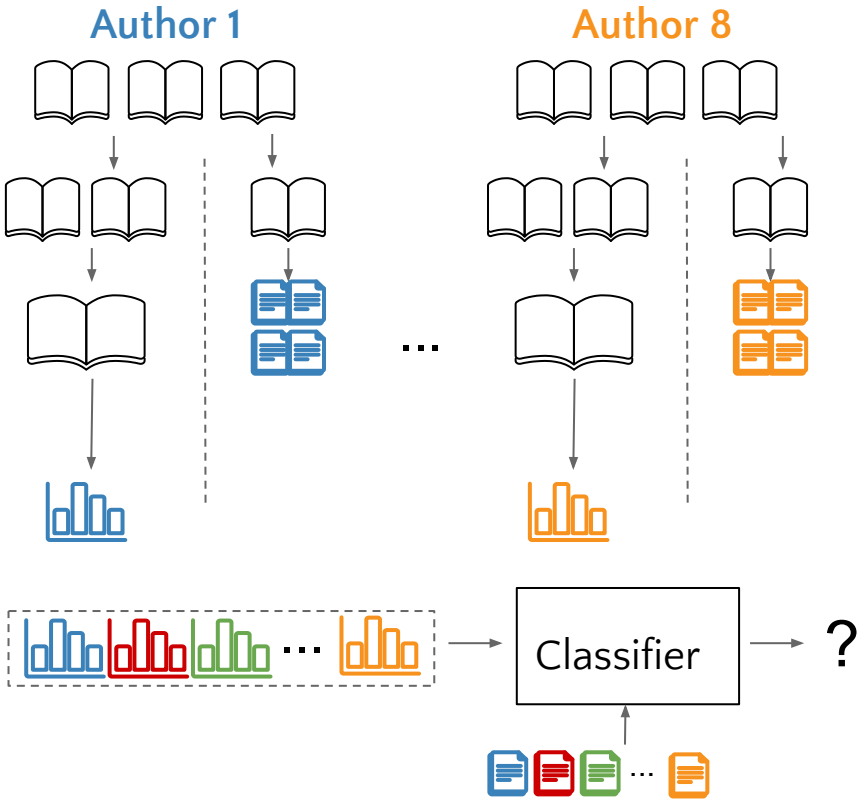
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** Test in chunks & label
- **Merge** Training books
- Obtain **Language Model**
- **Perplexity** based classifier

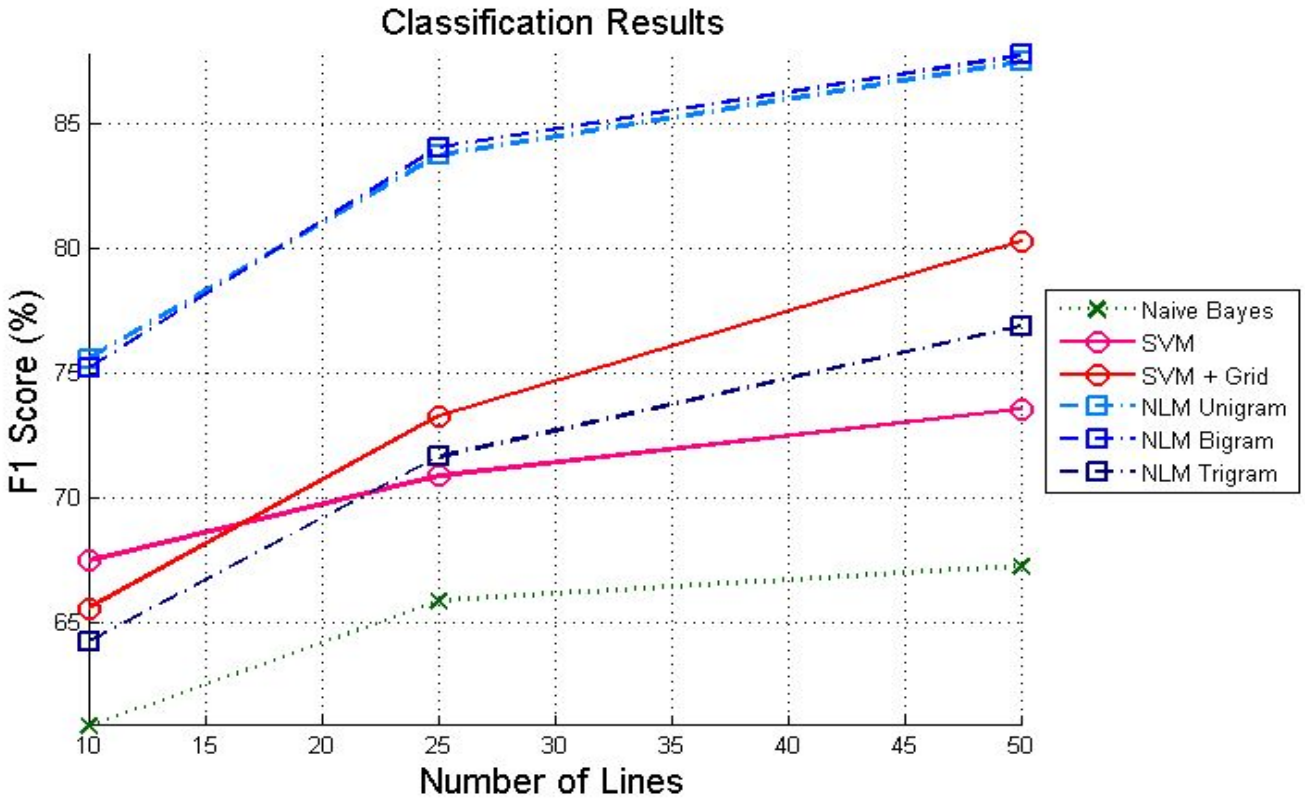


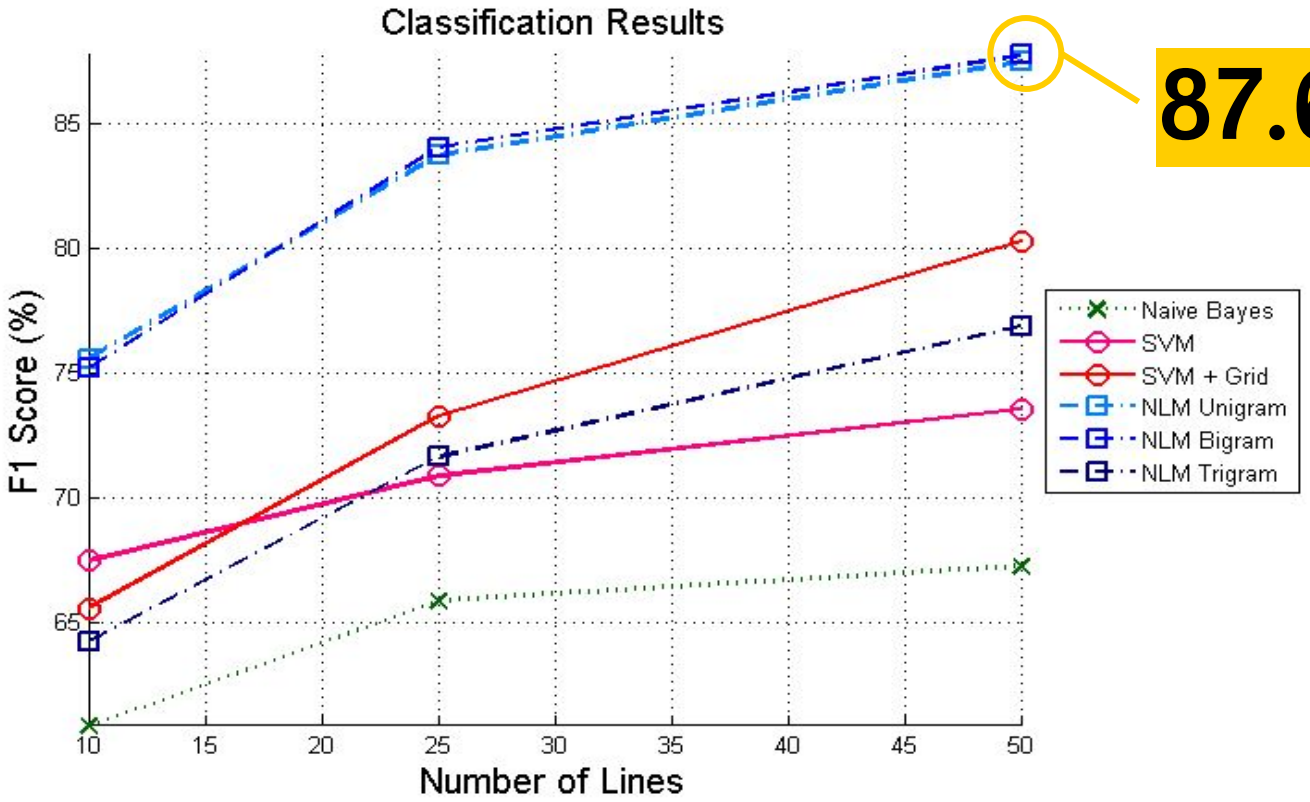
3 Approaches

- **Download** books
- 66% Training, 33% Test
- **Split** Test in chunks & label
- **Merge** Training books
- Obtain **Language Model**
- **Perplexity** based classifier
- Test Classifier



4 Results


Classification Results
F1 Score (%)
85
80
75
70
65
10
15
20
25
30
35
40
45
50
Number of Lines
Naive Bayes
SVM
SVM + Grid
NLM Unigram
NLM Bigram
NLM Trigram


Classification Results
F1 Score (%)
Number of Lines
85
80
75
70
65
10
15
20
25
30
35
40
45
50
Naive Bayes
SVM
SVM + Grid
NLM Unigram
NLM Bigram
NLM Trigram

87.6%

5 Conclusions. Future Work

5 Conclusions

- **NLM** is, by far, the **best option**
- **Larger chunks** (more lines) lead to **better results**
- **Correlation** represents a **problem** in simple approaches

5 Future Work

- **Increase** database (training and test)
- Further analyze **relation** between **chunk size** and **accuracy**
- **Equilibrate** number of samples per author
- **Explore** other **smoothing** methods of NLM
- Use of **Neural Networks**

Thanks!

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash

To evaluate the system

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

More Compact way:

$$\text{F}_1 \text{ Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$