NLTK use cases

- Remove stop words and persons' names in a **Recommendation System with NLTK**
- **Sentiment Analysis with NLTK** Check if the product review is positive or negative.
- **Build a chatbot using Python NLTK** Understand who is the target audience, and the intent or desire of the user and provide responses that can answer the user.

pros

- most well-known and full NLP library with many 3rd extensions
- supports the largest number of languages compared to other libraries

cons

difficult to learn and use
slow
only splits text by sentences, without analyzing the semantic structure
no neural network models

spaCy use cases

- **Search autocomplete** (and autocorect) is popular type of NLP that many people use on a daily basis.
- **Analyze online reviews**. Extract the key topics covered by the reviews without having to go through all of them. Help the sellers/retailers get consumer feedback in the form of topics (extracted from the consumer reviews).
- Automatic **Summarization of Resumes with NER** - Evaluate resumes at a glance to facilitate evaluation of resumes at a quick glance, thereby simplifying the effort required in shortlisting candidates among a pile of resumes.

pros

Fast
easy to learn and use

uses neural networks for training models

Cons

less flexibility compared to NLTK

Gensim use cases

- converting words and document to vectors
- finding text similarity
- text summarization

pros

- intuitive interface
- efficient implementation of popular algorithms
- scalable - can run latent semantic analysis and latent Dirichlet allocation on a cluster of computers

cons

- designed primarily for unsupervised text modeling
- don't implement full NLP pipeline, should be used with other libarary like Spacy or NLTK

1. **Text Preprocessing:**

- Best Fit: NLTK is well-suited for text preprocessing tasks such as tokenization, stop words removal, stemming, and lemmatization. It offers a wide range of functionalities and is easy to use for basic text cleaning and preprocessing.
- Worst Fit: Gensim is less suitable for text preprocessing compared to NLTK and spaCy. While it provides some functionalities for tokenization and text processing, it's primarily focused on higher-level NLP tasks like topic modeling and document similarity.

2. **Named Entity Recognition (NER):**

- Best Fit: spaCy excels in named entity recognition tasks. It provides pre-trained models with high accuracy for identifying and classifying named entities such as persons, organizations, and locations in text. SpaCy's efficient implementation makes it suitable for processing large volumes of text data.

- Worst Fit: NLTK's named entity recognition capabilities are not as advanced or efficient as spaCy's. While NLTK does offer some functionalities for NER, it may not perform as well or as quickly as spaCy for large-scale NLP tasks.

3. **Topic Modeling:**

- Best Fit: Gensim is considered one of the best libraries for topic modeling. It provides efficient implementations of popular algorithms like Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA). Gensim's distributed computing support also makes it suitable for processing large corpora and scaling up to big data.
- Worst Fit: NLTK and spaCy are less suitable for topic modeling compared to Gensim. While they offer some functionalities for text analysis and feature extraction, they are not specifically designed for topic modeling tasks and may lack the scalability and efficiency of Gensim for large-scale topic modeling projects.

4. **Sentiment Analysis:**

- Best Fit: NLTK offers various sentiment analysis tools, including pre-trained sentiment classifiers and lexicons like Vader, making it suitable for sentiment analysis tasks. It provides flexibility in building custom sentiment analysis pipelines and is widely used in research and education.
- Worst Fit: While spaCy and Gensim can be used for sentiment analysis, NLTK is generally preferred due to its dedicated sentiment analysis functionalities and extensive resources.

5. **Text Classification:**

- Best Fit: spaCy provides efficient implementations of text classification algorithms, including support vector machines (SVMs) and convolutional neural networks (CNNs). Its pre-trained models and customizable pipelines make it suitable for building text classification systems for various domains.
- Worst Fit: NLTK and Gensim can also be used for text classification, but they may lack the performance and efficiency of spaCy's models and tools for this task.

6. **Language Modeling:**

- Best Fit: Gensim excels in building and using language models such as Word2Vec and FastText. Its efficient implementations and support for distributed computing make it suitable for training large-scale language models on large corpora.
- Worst Fit: While NLTK and spaCy offer some functionalities for language modeling, they are not specifically designed for this task and may lack the scalability and efficiency of Gensim for training language models.

7. **Text Similarity:**

- Best Fit: spaCy provides efficient tools for computing text similarity based on word embeddings or document representations. Its pre-trained models and similarity metrics make it suitable for measuring similarity between texts for various applications such as recommendation systems and duplicate detection.
- Worst Fit: NLTK and Gensim can also be used for text similarity tasks, but they may not offer the same level of performance and ease of use as spaCy's tools and models.

8. **Document Summarization:**

- Best Fit: Gensim is well-suited for document summarization tasks, offering implementations of algorithms such as TextRank and LexRank for extracting key sentences or phrases from documents. Its support for topic modeling and document similarity also makes it useful for generating summaries based on document clusters.
- Worst Fit: While NLTK and spaCy can be used for document summarization, they may not provide the same level of efficiency and scalability as Gensim's algorithms and tools for this task.

9. Language Understanding and Dialogue Systems:

- Best Fit: spaCy offers efficient tools for building language understanding pipelines and dialogue systems. Its pre-trained models, support for custom pipelines, and integration with deep learning frameworks like TensorFlow and PyTorch make it suitable for developing advanced conversational AI systems.
- Worst Fit: While NLTK and Gensim can be used for language understanding tasks, they may not offer the same level of performance, efficiency, and integration capabilities as spaCy for building dialogue systems and conversational agents.

10. Text Messaging and Chatbots:

- Best Fit: spaCy is well-suited for building text messaging and chatbot applications. Its efficient parsing, named entity recognition, and part-of-speech tagging capabilities enable the development of conversational agents with natural language understanding.
- Worst Fit: While NLTK and Gensim can also be used for building chatbots, they may not offer the same level of performance and efficiency as spaCy's tools and models for this task.

11. Social Media Analysis:

- Best Fit: spaCy's named entity recognition and sentiment analysis capabilities make it suitable for analyzing social media data. Its pre-trained models and efficient processing enable the extraction of entities, sentiments, and trends from large volumes of social media content.
- Worst Fit: While NLTK and Gensim can also be used for social media analysis, they may not provide the same level of accuracy and efficiency as spaCy's tools and models for this task.

12. Personal Assistant Applications:

- Best Fit: spaCy is a good choice for building personal assistant applications such as scheduling assistants, task managers, and virtual assistants. Its natural language understanding capabilities, entity recognition, and parsing make it suitable for processing user commands and queries.
- Worst Fit: While NLTK and Gensim can be used for building personal assistant applications, they may require more effort and customization compared to spaCy due to their focus on lower-level NLP tasks.

13. Document Organization and Management:

- Best Fit: Gensim's topic modeling and document similarity functionalities make it suitable for organizing and managing documents. Its algorithms for topic extraction, document

clustering, and similarity retrieval enable the organization and categorization of large document collections.

- Worst Fit: While NLTK and spaCy can be used for document organization tasks, they may not offer the same level of efficiency and scalability as Gensim's algorithms and tools for this task.

14. Email Filtering and Organization:

- Best Fit: spaCy's named entity recognition and text classification capabilities make it suitable for email filtering and organization. Its pre-trained models and support for custom pipelines enable the automatic categorization and prioritization of emails based on content and sender.
- Worst Fit: While NLTK and Gensim can also be used for email filtering and organization, they may not provide the same level of accuracy and efficiency as spaCy's tools and models for this task.

Zoho Mail
avanish.batkulia@ai.datadisca.com
i?yhuDb5

Task -1

Write a python code to
1. Get a book Gutenberg from into a variable.
2. Remove the prefixes and postfixes including preface, content and references etc.
    1.1 How many words are there?
    1.2 How many puntuations are there and what are they?
    1.3 How many paragraphs are there?

3. Remove puntuations
4. Create a Pandas dataframe with the following columns
   Word, Frequency
5. What are the 50 most frequent words?
6. Given a prefix, write a funtion to return a frequecy dataframe with the prefix.

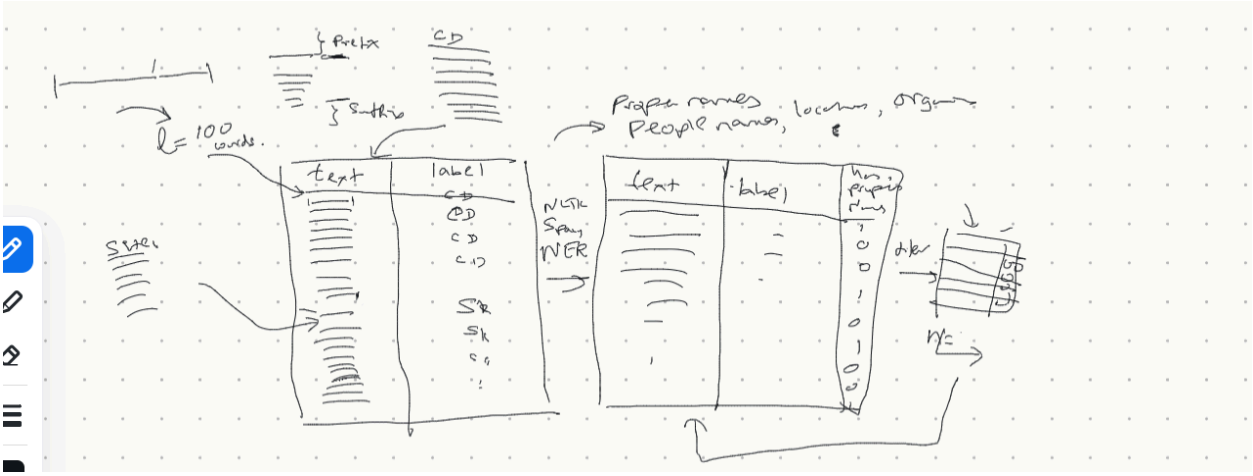
If a large number of rows involved, use only the most frequest 10,000 words.

Materials
https://github.com/DataDisca/nlp_author_identification_challenge
https://www.gutenberg.org/cache/epub/1023/pg1023.txt
https://www.machinelearningplus.com/nlp/gensim-tutorial/
https://www.geeksforgeeks.org/nlp-gensim-tutorial-complete-guide-for-beginners/

NO need to do preprocessing for most of the tasks, for e.g. in neural networks

For neural networks preprocessing is depends on the task in hand
Have to do lot of preprocessing in traditional models

TASK 2



NER(Name Entity Recognition)

All books have certain words like names, locations, and organization names that keep on repeating throughout the book. Thus it makes it easier for the model to predict the author based on these certain words, and thus model does not train to predict based on writing style.
Thus model might be able to perform well on the training set, but will not be able to perform well on the testing set, as now the model is not used to predict writing style but rather on entities, and each book has a different set of entities. Thus to counter this we remove the sequence of data from the book containing those entity names (here sequence refers to a set of 100 words or can say 100 grams like bi grams in n-grams).

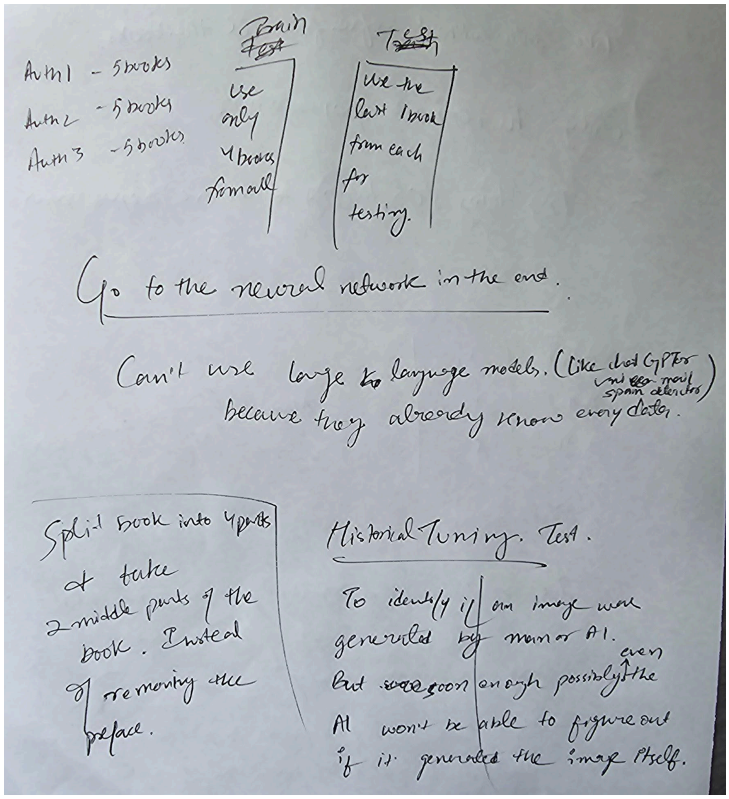
This thing is very often realised very later after we observe our model performing poorly on the testing set.

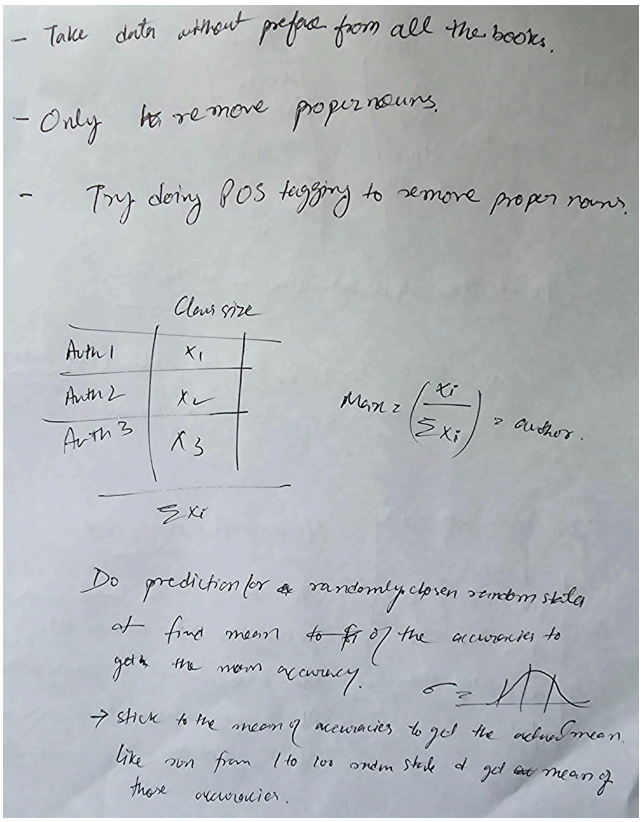
Steps:

1. Divide all data in the book to a set of certain sequence, for e.g. many set of 100 words and the label them with the author.
2. Then identify the entity in that book
3. Then list down which sequence of data contains an entity or not, no need to specify which entity only boolean output (0/1)
4. Remove the set of data containing entity
5. And then try to train the model on that data.

Q Do i remove sentences with any entity or a specific type of entity?
Ans. Yes remove only proper nouns, may use pos tagging to enhance it.

TASK3



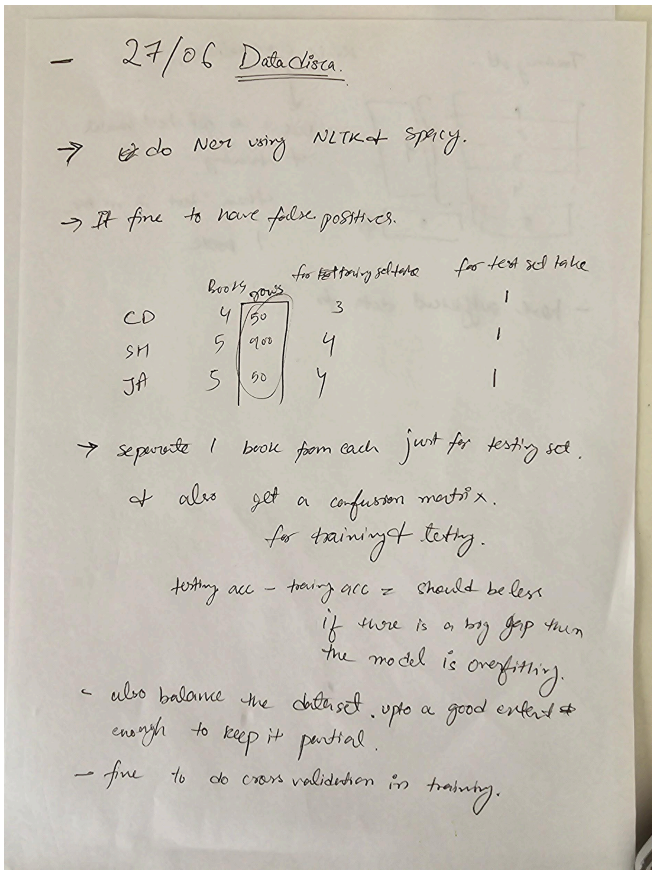


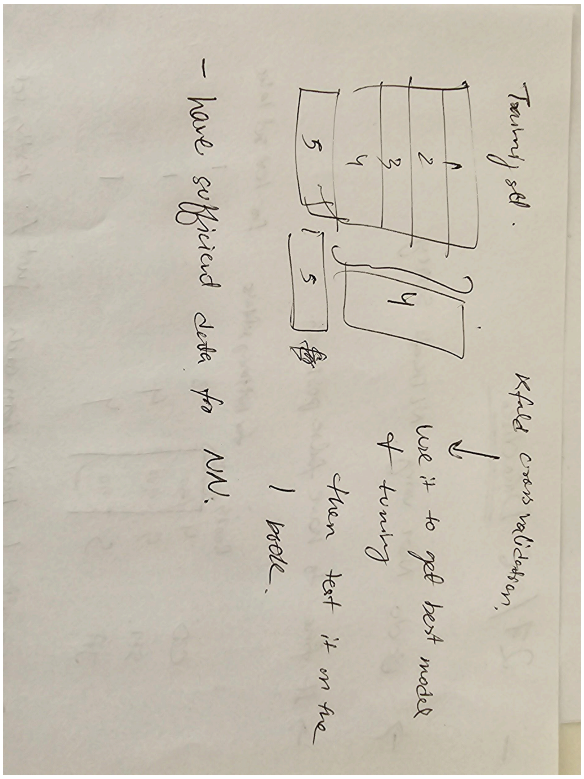
1. Remove preface data from all the book texts or just split the book into 4 parts and take the middle 2 parts.
    - *Can try using re and slice the book data from r'\bChapter\s+1\b'*
2. If we have taken 5 books from each author, use 4 books for training and the rest 1 for testing. So it helps us evaluate our model more accurately.
3. To get the actual accuracy of our model instead of doing it on a single random state, do it on multiple randomly chosen random states and get the mean of the accuracy of the accuracies of the random states.
    - *Try using k-fold cross validation*
4. For removing entities with proper nouns from the data set use pos tagging to enhance entity identification in the data set. Remove only proper nouns.
5. In the end try using the neural networks.

- *Can't use Bert or gpt3 like large language models because they already have the knowledge of which text belongs to which author.*

Q. How can i use pos tagging to improve ner ?

TASK-4





Perform ner before pre processing because spacy need context of the sentences to perform ner properly.