# Text Dataset Comparison: BoW, Similarity, and Library Performance

Victor Hugo Figueroa
Kristiania University
vifi001@student.kristiania.no

Alin-Eugen Gusanu
Kristiania University
algu008@student.kristiania.no

Hashem Sheikh
Kristiania University
hash011@student.kristiania.no

Jesse Lang
Kristiania University
jela019@student.kristiania.no

## 1 Introduction

This paper's focus is exploring the connections between three distinct datasets. The datasets explored were descriptions of cities in Norway, the classic novel Crime and Punishment by Fydor Dostoevsky and a dataset regarding text mining (NLP). Notably, these datasets are dissimilar from one another. To analyze and identify common themes and patterns for the different texts, text mining techniques, such as Bag of Words (BoW) were used.

The project´s aim revolves around the assessment and development of a Bag of Words (BoW) model to gain insights from text data. We compare the performance and similarity of our BoW implementation and the implementation in scikit-learn. The evaluation will outline the strengths and weaknesses of different implementations of BoW and how they can highlight meaningful information from the texts.

Can the application of Bag of Words (BoW) representations and similarity measurements help identify common word sets and differences across text datasets about Norway, text mining, and books, and how does the performance of established libraries like sklearn compare to custom implementations in this context?

## 2 Methodology

For the project completion multiple libraries have been used with Python 3 version 3.10.6. The main libraries used are pandas, numpy, sklearn, nltk and matplotlib.

### 2.1 Pandas

Pandas is a Python package that provides "fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language" (pandas.pydata.org, 2023).

### 2.2 NumPy

NumPy is a Python library for scientific computing, it provides a high-performance multidimensional array object and a collection of routines for operating on arrays. NumPy is used in a wide variety of scientific fields, including physics, engineering, and finance. Another key benefit of NumPy is its flexibility. NumPy arrays can be used to represent a variety of data types, including integers, floats, strings, and complex numbers. This makes NumPy a good choice for a wide range of scientific computing tasks (Harris et al., 2020).

### 2.3 Scikit-learn

Scikit-learn is a popular Python library for machine learning (ML) tasks, including classification, regression, clustering, and dimensionality reduction (Pedregosa et al., 2011). It is designed to be user-friendly and efficient, with a consistent interface for all algorithms and utilities for data preparation, model evaluation, and deployment.

### 2.4 NLTK

Natural Language Toolkit (NLTK) is a free, open-source Python library for natural language processing (NLP) (Bird et al., 2009). It provides a wide range of NLP tasks, including tokenization, stemming, lemmatization, named entity recognition, parsing, and machine translation (nltk.org, 2023).

NLTK is one of the most popular NLP libraries for Python, and it is used by researchers and practitioners around the world.

NLTK is designed to be easy to use and efficient. It provides a simple and consistent interface for all its tasks, making it easy to get started with NLP in Python. NLTK also provides several pretrained NLP models, which can be used to perform common NLP tasks without having to train a model from scratch (nltk.org, 2023). NLTK is a powerful tool for NLP in Python. It is suitable for a wide range of tasks, from simple text classification to complex machine translation models.

### 2.5 Matplotlib

Matplotlib is a free, open-source Python library for data visualization. It provides a wide range of plotting functions, including line plots, bar charts, histograms, and scatter plots. Matplotlib is one of the most popular data visualization libraries for Python, and it is used by researchers and practitioners around the world. Matplotlib is designed to be flexible and powerful. It can be used to create a wide variety of data visualizations, from simple line plots to complex interactive charts. Matplotlib also provides several features for customizing the appearance of data visualizations, such as fonts, colours, and labels (matplotlib.org, 2023).

### 2.6 Pipeline

Figure 1 is representing the pipeline used in this project. It shows the steps from the cleaning process until the end outcome which prints the similarities between the given input. The first step is to preprocess the given text data, it needs to be cleaned from null values and split into a character/word/n-gram, so called tokens (Lane, H., 2019). Those tokens often contain some noise that is unwanted to analyze the data, therefore whitespace and stop words need to be removed and only alpha numerical values kept.

A function iterates over each token in the given list and counts the occurrence of each item and stores it with the token in a dictionary. After the function it has created a Bag of Words for each of the given text inputs. A BoW (Bag of Words) is a dictionary of tokens with a counter of each recurrent value 'Good': 1, 'Rosa!': 1, 'morning,': 1 (Lane, H., 2019).

The next step would be the creation of a corpus (list of documents) and vectorizing the data. The previously created Bag of Words will be one hot encoded and saved into a list based on their term frequency (see figure 1, step 3).

Lastly, the calculation of the similarities between the different texts in the corpus is done. One of the ways this could be achieved is by calculating the cosine similarity.

$$CosineSimilarity(A,B) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

This is the cosine of the angle between vector A and vector B. It is an efficient way since it does not require any trigonometric evaluations and has a convenient range for most problems (-1 to +1) (Lane, H., 2019).

## 3 Results

| Method | LOC | Total |
|---|---|---|
| sklearn | 4 import + 12 code | 16 |
| tokenization method | 6 import + 20 functions + 24 code | 50 |

Table 1: Cosine similarity of the different datasets

The result of the tokenization method took time to implement as seen in figure 1. It took 6 imports and 20 functions with 24 lines of code. Using sklearn was much quicker to implement. The sklearn method could do it in 12 lines of code and 4 imports.

In visualizing the BoW figure 2 (left) the top used words in the data seem to originate from the Crime and Punishment book. Figure 2 (right) depicts the data when a Tf-idf is used instead of BoW shifting the weight from number of times used to rarer used words.

Figure 3 depicts how BoW might now be the best in this use case, by having CP hold 99.88 percent of the data.

| | tromso | oslo | nlp | CP |
|---|---|---|---|---|
| tromso | 1.000000 | 0.121268 | 0.000000 | 0.008290 |
| oslo | 0.121268 | 1.000000 | 0.000000 | 0.008098 |
| nlp | 0.000000 | 0.000000 | 1.000000 | 0.026442 |
| CP | 0.008290 | 0.008098 | 0.026442 | 1.000000 |

Table 2: Cosine similarity with sklearn

| | tromso | oslo | nlp | CP |
|---|---|---|---|---|
| tromso | 1.000000 | 0.111154 | 0.0000 | 0.008419 |
| oslo | 0.111154 | 1.000000 | 0.0000 | 0.010073 |
| nlp | 0.000000 | 0.000000 | 1.0000 | 0.026400 |
| CP | 0.008419 | 0.010073 | 0.0264 | 1.000000 |

Table 3: Cosine similarity with tokenization method

Above table 3 is the singularity matrix of the BoW using the tokenization method, and table 2 is the singularity matrix of the sklearn method. The figures above show that the NLP data shares no similarities with the Norway data, but it does share a bit with the book data.

# 4 Discussion

For the data analysis, as can been seen from the pipeline (figure 1), the Bag of Words (BoW) model was used, and two distinct deployment approaches were implemented. One approach involved the traditional, more time-consuming method, which required the utilization of tokenization and the NLTK library. As in the pipeline visualized are there many different steps required for this implementation, this could lead to more bugs and errors when running the code. The other method was using sklearn to do all the processing, therefore two slightly different results were outlined (Table 2 and Table 3). The difference might come from the fact that sklearn has a much larger stop words list that helps in reducing possible noise in the data. While sklearn is a faster way to deploy BoW, the tokenization approach does give a more customizable analysis.

By further analyzing Table 2 and Table 3 it became very clear that it had another intriguing insight to mine. When examining the 'NLP' text, it becomes clear that it shares little to no values with the other texts. In fact, the only noticeable intersection it had was with 'CP,' bordering on just 2% similarity. It's worth noting that 'CP' is a large book compared to the shorter texts, underscoring the uniqueness of the 'NLP' text. This discovery helped in emphasizing the limited similarities the book shares with the other texts, with none even reaching 3% similarity.

After the review of the BoW graph, the following fact came out, the name of the main character had clearly taken over the rest of the words with almost 800 instances, followed by "one"," would", and "know". Therefore, the results were not satisfactory, and further investigation of the BoW model was conducted. After comparing the size of the data, it was clear who was the culprit for these results. Out of all the data, Crime and Punishment had an astonishing 99.88% of the size, therefore leaving the other texts with a cumulative value of 0.12%.

As a solution to this problem, the Tf-idf model was used. The Tf-idf model has a different approach to the problem, as it weighs rarer words higher than the words with a high occurrence. Therefore, offering the possibility of reducing the noise that the book created. Figure 2 (right) illustrates the Tf-idf model, showing that it effectively mitigates the influence of excessively frequent terms and improves the overall balance in the analysis.

# 5 What we learned

The teamwork that was required for the completion of this project provided us with meaningful experiences, as well as a better understanding of different methods to analyse texts. It gave us the chance to explore, learn, and try out different approaches for the same problem. Moreover, we have got a glimpse of how to work in a team within a professional environment.

The importance of picking the right tools for the job was highlighted after the analysis of the different approaches. The tokenization method gave us the freedom to tweak the parameters for optimising the outcome, whereas the already built models from sklearn won in terms of efficiency, speed, and noise reduction due to its larger list of stopwords.

Lastly, the whole assignment was game changing, offering hands-on experience on how to work with the input you get. It has enhanced our skills and sharpened our vision on working with text data, it made us go beyond and look deeper into the data and how to manipulate and work with it.

# References