Using Machine Learning To Find Inventions in Science Fiction

Mateusz Woźniak

1. Introduction

Upon looking up the term “science fiction” in Google, the very first definition presented is: “fiction based on imagined future scientific or technological advances and major social or environmental changes, frequently portraying space or time travel and life on other planets.”. By definition this genre incorporates some sort of vision of the future and fascinating inventions described by the author. How can we find the inventions, machines and devices without spending thousands of hours reading hundreds of books? Did their ideas materialize in the future?

2. Goal of the project

Main aim of the project is to propose a framework that could serve a baseline for finding science fiction writers that predicted future inventions to some extent. Prediction of the future itself can be interpreted in variety of ways, therefore project’s focus is on providing perspective and approach on the topic rather than solving and exhausting it. Framework provided should use computational methods applied for literature analysis.

Second goal of the project is to tell a story of timeless ideas of few science fiction writers and bring humanities to the digital in a visual form of a poster.

3. Methodology

First and foremost issue of the project is how to extract invention from science fiction books without spending thousands of hours reading hundreds of books. In the year 2000 Franco Moretti coined the term “distant reading” referring to the analysis of literature becoming a patchwork of of other’s peoples research. (Moretti, 2014). Later due to his and other scholars’ work this term became a description of computational analysis in literature.

First and foremost to complete the project, it is necessary to identify the inventions from science fiction books. The source are the books themselves that were scraped from the Project Gutenberg (“Project Gutenberg,” n.d.). From each book particular sentences were isolated based on keywords such as “machine”, “device” or “invention”. This by itself is not enough to know if the paragraph is a description of mentioned machine or device, or just a particular word just happened to be a part of a dialogue between two characters, or maybe a keyword was used in a metaphor. To figure out which sentences are describing a device, a logistic regression based (“Logistic Regression - an overview | ScienceDirect Topics,” n.d.) classifier was trained. Library used was scikit-learn for Python. (“scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation,” n.d.) Logistic regression was used in this application because of a binary nature of the text: it is either a description of a device or not. Logistic regression in its nature takes two values and fits the logistic S-curved function in between. This statistical method allows to predict the outcome of the test by one factor. In case of this project’s classifier the factor is the labelled text used as a training data. There were two data sources used to train the model. Firstly books from genres of fantasy, romance and crime were scraped for paragraphs and labelled as “book”. Secondly patents abstracts were

acquired from Google Patents Public Data (“Google Patents Public Data – Marketplace – DH2023 – Google Cloud Console,” n.d.), which was free to access at the time of this project. Data was retrieved using BigQuery and labelled as “invention”. Trained classifier was used to evaluate mined paragraphs from science fiction genre to identify descriptions of inventions, devices, machines, etc.

4. Technical framework

First script used is the web scraper. Script is importing four libraries necessary to read and write CSV files, communicate with Project Gutenberg, and scrape the website. The HTML parser used is Beautiful Soup 4 (“Beautiful Soup Documentation — Beautiful Soup 4.12.0 documentation,” n.d.). Scraping of Project Gutenberg is a straight forward process. Script is receiving a URL address of a page of a selected genre, all the anchor elements are looked up on the website. From each of the gathered elements, last digits of the address are the book IDs, these are extracted and collected in an array. Based on the IDs books are being downloaded as text files, read and searched for keywords: “device”, “machine”, “invention” and “contraption”. Script is saving the sentence in which the keyword was found together with previous and next sentence to extract longer fragments and avoid extracting large paragraphs. Together with the sentence, to the CSV file are written the sentence number and book ID. Books and CSV file are saved to the local machine.

This script is used to collect training data for classifier, and science-fiction data that is later classified. Code can be found in a raw form in appendix 1.

Second script used is the classifier (appendix 2). Classifier is using scikit-learn library for Python and logistic regression described in the methodology section. At first script is loading three required files. First file needed is the books fragments extracted from genres other than science fiction extracted by the web scraper. This data will train the classifier. Second file required is the patent abstracts data, which is extracted from Google’s database with a simple query (appendix 3) and later downloaded as a CSV. Finally, the last file needed are the fragments of science fiction books to be classified. Training data is being labelled as either “book” or “invention”, and two files are combined. Later text is being vectorized using TF-IDF, which is popular technique using term frequency and inverse document frequency to calculate weight of the words. Next classifier is trained and the results are presented in console.

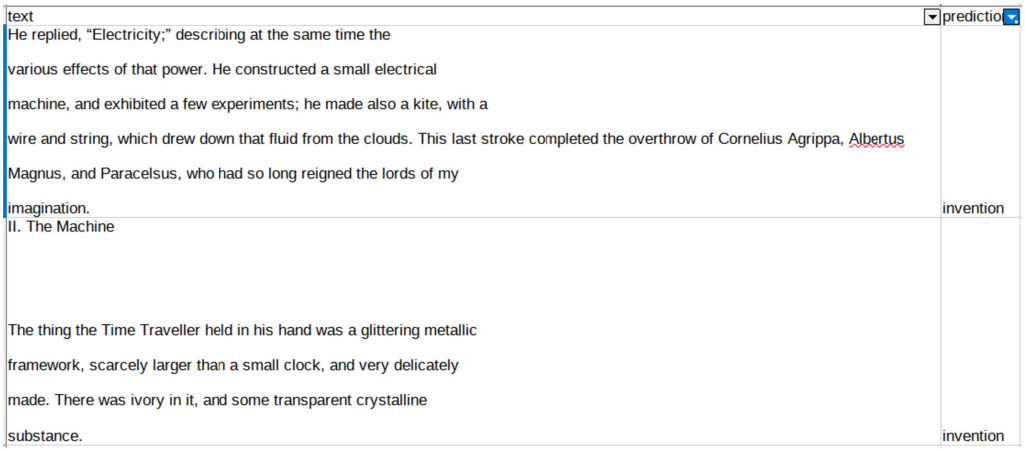
```
In [17]: runfile('C:/Users/Matt/Desktop/DH_FinalProject/untitled0.py', wdir='C:/Users/Matt/
Desktop/DH_FinalProject')
Accuracy on validation set: 0.9912790697674418
Classification Report:
              precision    recall  f1-score   support

        book       1.00      0.98      0.99       147
      patent       0.98      1.00      0.99       197

    accuracy                           0.99       344
   macro avg       0.99      0.99      0.99       344
weighted avg       0.99      0.99      0.99       344
```

Figure 1: Evaluation of the model

Finally, science fiction fragments are classified, results are presented in console and saved to a CSV file.



text	predictio
He replied, “Electricity;” describing at the same time the various effects of that power. He constructed a small electrical machine, and exhibited a few experiments; he made also a kite, with a wire and string, which drew down that fluid from the clouds. This last stroke completed the overthrow of Cornelius Agrippa, Albertus Magnus, and Paracelsus, who had so long reigned the lords of my imagination.	invention
II. The Machine The thing the Time Traveller held in his hand was a glittering metallic framework, scarcely larger than a small clock, and very delicately made. There was ivory in it, and some transparent crystalline substance.	invention

Figure 2: Results

As shown above, descriptions of little electrical machine and Time Traveller’s device are correctly predicted to be an invention. Both fragments describe either the look or a usage of an invention similarly to how an abstract of a patent would be formed.

5. Issues and solutions

Presented fragment of the results shows positive outcome from the classifier, however the model will always be as good, as the data used for training it. In case of this demonstration small amounts of data were used. Web scraper mined only 355 fragments from one page of Science Fiction. Classifier was trained on 14333 fragments of books from genres of fantasy, romance and crime, together with 13776 abstract of patents. For more precise results more data can be used. Furthermore, patents data was not filtered for any specific category, so the model was trained on patents of engineering inventions together with biomedical breakthroughs.

Web scraper could be potentially improved to check for the book title and other information on the Project Gutenberg website to replace book IDs with titles, add authors name to the file and year in which the book was published to avoid manual work.

6. Few words on storytelling

Steps taken allowed to prepare a list of invention descriptions from science fiction books accessible at Project Gutenberg’s website. Such list is easier to analyse to see whether described inventions happened to become a real invention in the future and when did it happen in comparison to when the book was written. Together with each paragraph other information can be retrieved such as books title, year of publishing and author. Using other methods of distant reading such as Named-entity recognition (NER) allowed to isolate places from the book (if they were actual places on the Earth), and compare them with places where author of the book lived or stayed. By compiling all the information captured, a story of authors and their ideas can be created. This paper is focusing on application of machine learning in Digital Humanities to isolate descriptions of inventions from Science Fiction books. Paper is supplemented by a poster to be presented during the further evaluation of the project.

7. Bibliography

Beautiful Soup Documentation — Beautiful Soup 4.12.0 documentation [WWW Document], n.d. URL https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed 1.4.24).

Google Patents Public Data – Marketplace – DH2023 – Google Cloud Console [WWW Document], n.d. URL https://console.cloud.google.com/marketplace/product/google_patents_public_datasets/google-patents-public-data?project=dh2023-409213 (accessed 1.3.24).

Logistic Regression - an overview | ScienceDirect Topics [WWW Document], n.d. URL https://www.sciencedirect.com/topics/computer-science/logistic-regression (accessed 1.3.24).

Moretti, F., 2014. “Conjectures on World Literature.” New Left Rev. 1, 54–68.

Project Gutenberg [WWW Document], n.d. . Proj. Gutenberg. URL https://www.gutenberg.org/ (accessed 1.3.24).

scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation [WWW Document], n.d. URL https://scikit-learn.org/stable/ (accessed 1.3.24).

8. Appendix 1

Web scraper script

```
import csv
import os
import nltk
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize

def get_book_ids_from_url(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        links = soup.find_all('a', class_='link')
        book_ids = []
        #print(links)

        for link in links:
            href = link.get('href', '')
            #print(href)
            book_id = href.split('/')[-1]
            #print(book_id)
            if book_id.isdigit():
                book_ids.append(int(book_id))

        return book_ids

    return []
```

Part 1

```python
def download_gutenberg_books(book_ids, output_folder):
    for book_id in book_ids:
        url = f'http://www.gutenberg.org/cache/epub/{book_id}/pg{book_id}.txt'
        response = requests.get(url)

        if response.status_code == 200:
            text = response.text
            output_file_path = os.path.join(output_folder, f"{book_id}.txt")

            with open(output_file_path, 'w', encoding='utf-8') as file:
                file.write(text)

def extract_books_sentences(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        sentences = nltk.sent_tokenize(file.read())

    # Keywords to look for in sentences
    keywords = ['device', 'machine', 'invention', 'contraption']

    # Placeholder for books descriptions and page numbers
    books_data = []

    # Add logic to identify and extract books descriptions
    for i, sentence in enumerate(sentences, start=1):

        if any(keyword in sentence.lower() for keyword in keywords):
            # Extract the previous and next sentences
            prev_sentence = sentences[i - 2] if i >= 2 else ""
            next_sentence = sentences[i] if i < len(sentences) else ""

            # Combine the sentences and add to device_data
            combined_sentences = f"{prev_sentence} {sentence} {next_sentence}".stri
            books_data.append({'Page': i, 'Description': combined_sentences})

    return books_data
```

Part 2 (method called outside of the screenshot is .strip())

```python
def save_to_csv(books_data, csv_file_path, book_title):
    # Check if the CSV file already exists
    file_exists = os.path.exists(csv_file_path)

    with open(csv_file_path, 'a', newline='', encoding='utf-8') as csvfile:
        fieldnames = ['Book', 'Page', 'Description']
        csv_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        # Write header if the file is newly created
        if not file_exists:
            csv_writer.writeheader()

        # Write data to CSV file
        for data in books_data:
            data['Book'] = book_title
            csv_writer.writerow(data)

# Required URLs
gutenberg_url = r'https://www.gutenberg.org/ebooks/subject/1065'
output_folder = r'C:\Users\Matt\Desktop\DH_FinalProject\output'
output_csv_path = r'C:\Users\Matt\Desktop\DH_FinalProject\output\output_training.cs

# Get book IDs from the specified URL
book_ids = get_book_ids_from_url(gutenberg_url)

# Download books and process them
for book_id in book_ids:
    download_gutenberg_books([book_id], output_folder)
    book_title = f"Gutenberg_{book_id}"
    book_file_path = os.path.join(output_folder, f"{book_id}.txt")
    books_data = extract_books_sentences(book_file_path)

    # Save the extracted device descriptions with page numbers to the CSV file
    save_to_csv(books_data, output_csv_path, book_title)

print(f"Books sentences with enumeration saved to {output_csv_path}")
```

part 3

Appendix 2

Classifier

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Prepare the Data

# Load patent descriptions
patent_data = pd.read_csv(r'C:\Users\Matt\Desktop\DH_FinalProject\patent_training.cs

# Load mined paragraphs for training
mined_data = pd.read_csv(r'C:\Users\Matt\Desktop\DH_FinalProject\output\output_train

# Load mined paragraphs from scifi
scifi_mined = pd.read_csv(r'C:\Users\Matt\Desktop\DH_FinalProject\output\outputauto.

# Step 2: Label the Data

patent_data['label'] = 'invention'
mined_data['label'] = 'book'

# Step 3: Combine and Shuffle Data

combined_data = pd.concat([patent_data, mined_data], ignore_index=True)
combined_data = combined_data.sample(frac=1, random_state=42).reset_index(drop=True)

# Step 4: Text Vectorization

vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X = vectorizer.fit_transform(combined_data['text'].values.astype('U'))
y = combined_data['label']
```

Part 1

```
# Step 5: Train a Classifier

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=

classifier = LogisticRegression()
classifier.fit(X_train, y_train)

# Step 6: Evaluate the Model

y_pred = classifier.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)

print(f"Accuracy on validation set: {accuracy}")
print("Classification Report:")
print(classification_report(y_val, y_pred))

# Step 7: Classify Mined Paragraphs

mined_paragraphs = vectorizer.transform(scifi_mined['text'])
mined_predictions = classifier.predict(mined_paragraphs)

scifi_mined['prediction'] = mined_predictions

# Display the results
print("\nClassified Mined Paragraphs:")
print(scifi_mined[['text', 'prediction']])

results_csv_path = r'C:\Users\Matt\Desktop\DH_FinalProject\results_training.csv'
combined_data.to_csv(results_csv_path, index=False)
scifi_mined.to_csv(r'C:\Users\Matt\Desktop\DH_FinalProject\mined_results_scifi.csv',

print(f"\nResults saved to '{results_csv_path}' and 'path/to/mined_results.csv'.")
```

Part 2

Appendix 3

Google BigQuery

```
1  SELECT DISTINCT abstract FROM `patents-public-data.patentsview.patent` LIMIT 100000
```