**# Dataset Setup Guide for Plagiarism Detection System**

This guide will help you download, preprocess, and integrate datasets into your plagiarism detection system.

**## 🚀 Quick Start (3 Simple Steps)**

**### Step 1: Download Datasets**

```bash

python dataset\_downloader.py

```

Choose option 4 to download all datasets, or select specific ones.

**### Step 2: Preprocess Data**

```bash

python dataset\_preprocessor.py

```

This cleans and prepares the data for training.

**### Step 3: Integrate into Django**

```bash

python dataset\_integrator.py

```

This adds the data to your Django database and trains TF-IDF models.

**## 📊 What You'll Get**

**### \*\*Reference Documents\*\***

- Added to `ReferenceDocument` model for local plagiarism checking

- Used by "Check File" and "Check Plagiarism" functions

- Provides baseline comparison for similarity detection

**### \*\*Trained Models\*\***

- TF-IDF vectorizers trained on different datasets

- Stored in `TrainedDatasetModel` for custom similarity checking

- Improves accuracy of plagiarism detection

**### \*\*Dataset Documents\*\***

- Stored in `DatasetDocument` model for training and evaluation

- Includes both original and artificially plagiarized content

- Used for model training and testing

**## 🎯 Dataset Sources Included**

**### \*\*1. Wikipedia Articles\*\***

- **\*\*Content\*\***: 100 random Wikipedia articles

- **\*\*Use\*\***: General knowledge reference corpus

- **\*\*Type\*\***: Original, non-plagiarized content

**### \*\*2. ArXiv Papers\*\***

- **\*\*Content\*\***: 50 recent computer science paper abstracts

- **\*\*Use\*\***: Academic reference corpus

- **\*\*Type\*\***: Original research content

**### \*\*3. Artificial Plagiarism Samples\*\***

- **\*\*Content\*\***: Paraphrased, reordered, and partial copies

- **\*\*Use\*\***: Training plagiarism detection algorithms

- **\*\*Type\*\***: Labeled plagiarized content

**## ⚙️ Technical Details**

**### \*\*Text Processing\*\***

- Unicode normalization (NFKC)

- Whitespace cleaning and normalization

- Text chunking (300-500 words per chunk)

- HTML/XML tag removal

**### \*\*TF-IDF Training\*\***

- Max features: 10,000

- N-grams: 1-3 (unigrams, bigrams, trigrams)

- Stop words: English

- Min document frequency: 2

- Max document frequency: 95%

**### \*\*Model Storage\*\***

```

trained\_models/

├── combined\_dataset/

│   ├── tfidf\_vectorizer.pkl

│   └── tfidf\_matrix.pkl

├── wikipedia\_dataset/

│   ├── tfidf\_vectorizer.pkl

│   └── tfidf\_matrix.pkl

└── arxiv\_dataset/

    ├── tfidf\_vectorizer.pkl

    └── tfidf\_matrix.pkl

```

**## 🔧 Customization Options**

**### \*\*Add Your Own Data\*\***

1. Create JSON file with this format:

```json

[

  {

    "id": "custom\_1",

    "title": "Document Title",

    "content": "Document content text...",

    "source": "your\_source",

    "type": "original"

  }

]

```

2. Place in `processed\_datasets/` folder

3. Run `dataset\_integrator.py`

**### \*\*Modify Processing\*\***

- Edit `dataset\_preprocessor.py` to change:

  - Text cleaning rules

  - Chunk sizes

  - Paraphrasing methods

  - Data sources

**### \*\*Adjust TF-IDF Parameters\*\***

- Edit `dataset\_integrator.py` to modify:

  - Feature count (`max\_features`)

  - N-gram range (`ngram\_range`)

  - Document frequency thresholds

**## 🧪 Testing Your Setup**

After integration, test your system:

1. **\*\*Check Plagiarism\*\*** - Should use new reference documents

2. **\*\*Check File\*\*** - Should work with enhanced reference corpus

3. **\*\*Custom Models\*\*** - Should show improved accuracy

**## 📈 Expected Improvements**

With proper dataset integration:

- **\*\*Higher accuracy\*\*** in plagiarism detection

- **\*\*Better coverage\*\*** of different text types

- **\*\*Reduced false positives\*\*** through diverse training data

- **\*\*Improved similarity calculations\*\*** with TF-IDF models

**## 🔍 Troubleshooting**

**### \*\*Common Issues\*\***

- **\*\*Django import errors\*\***: Ensure you're in the project directory

- **\*\*Memory issues\*\***: Reduce dataset size or chunk size

- **\*\*Model training fails\*\***: Check if enough documents (minimum 10)

- **\*\*Database errors\*\***: Ensure Django database is properly configured

**### \*\*Debug Commands\*\***

```python

# Check database contents

python manage.py shell

>>> from plagiarismchecker.models import \*

>>> print(f"Reference docs: {ReferenceDocument.objects.count()}")

>>> print(f"Trained models: {TrainedDatasetModel.objects.count()}")

>>> print(f"Dataset docs: {DatasetDocument.objects.count()}")

```

**## 📝 Next Steps**

1. Run the 3-step setup process

2. Test plagiarism detection with known examples

3. Monitor accuracy and adjust parameters as needed

4. Add more datasets over time to improve performance

---

**\*\*Need help?\*\*** Check the debug output from each script for detailed information about the process.