Environment Setup

```
# Core parsing, NLP, ML, and plotting
!pip -q install beautifulsoup4 lxml textstat scikit-learn sentence-
transformers numpy pandas tqdm nltk unidecode rapidfuzz requests
seaborn matplotlib wordcloud plotly

# Advanced NLP for NER and Topic Modeling
!pip -q install spacy bertopic umap-learn hdbscan

# Scalable lexical near-duplicate detection
!pip -q install datasketch

from bs4 import BeautifulSoup

# NLTK resources for tokenization & sentiment
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')  # newer NLTK tokenizers (some Colab images
require this)
nltk.download('vader_lexicon')

# Load spaCy English model (small) with fallback download in Colab
import spacy, spacy.cli as spcli
try:
    nlp = spacy.load("en_core_web_sm")
except:
    spcli.download("en_core_web_sm")
    nlp = spacy.load("en_core_web_sm")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Folders

```
import os, random
import numpy as np

# Create the expected repository-like structure in Colab VM
base = "seo-content-detector"
os.makedirs(f"{base}/data", exist_ok=True)
os.makedirs(f"{base}/models", exist_ok=True)
os.makedirs(f"{base}/notebooks", exist_ok=True)
os.makedirs(f"{base}/streamlit_app/utils", exist_ok=True)
os.makedirs(f"{base}/streamlit_app/models", exist_ok=True)
print("Folders prepared:", os.listdir(base))
```

```python
# Set deterministic seeds for reproducibility in ML steps
def set_seeds(seed: int = 42):
    random.seed(seed)
    np.random.seed(seed)
    try:
        import torch
        torch.manual_seed(seed)
        torch.cuda.manual_seed_all(seed)
    except Exception:
        pass

set_seeds(42)

# Log simple environment metadata to help debug version drift across
runs
import platform, sys, subprocess
print("Python:", sys.version)
print("Platform:", platform.platform())

# Snapshot key pip packages
!python -m pip freeze | grep -E "scikit-learn|pandas|numpy|sentence-
transformers|spacy|bertopic|umap-learn|hdbscan" || true
```

```
Folders prepared: ['notebooks', 'requirements.txt', 'README.md',
'models', 'data', 'streamlit_app', '.gitignore']
Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
Platform: Linux-6.6.105+-x86_64-with-glibc2.35
bertopic==0.17.3
en_core_web_sm @
https://github.com/explosion/spacy-models/releases/download/en_core_we
b_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-
any.whl#sha256=1932429db727d4bff3deed6b34cfc05df17794f4a52eeb26cf8928f
7c1a0fb85
geopandas==1.1.1
hdbscan==0.8.40
numpy==2.0.2
pandas==2.2.2
pandas-datareader==0.10.0
pandas-gbq==0.29.2
pandas-stubs==2.2.2.240909
scikit-learn==1.6.1
sentence-transformers==5.1.2
sklearn-pandas==2.2.0
spacy==3.8.7
spacy-legacy==3.0.12
spacy-loggers==1.0.5
umap-learn==0.5.9.post2
```

Load Dataset and HTML Parsing + Parsing Visuals

```
import pandas as pd
import re
from bs4 import BeautifulSoup
from unidecode import unidecode

# Load provided primary dataset: must have url and html_content
columns
df = pd.read_csv("/content/seo-content-detector/data/data.csv")
assert {'url','html_content'}.issubset(df.columns), "Expected columns:
url, html_content"

# Heuristic main content extractor: prefer <main>/<article>; strip
boilerplate tags
def extract_main_text(html: str) -> str:
    """
    Extract the main body text from a page:
    - Prefer <main> or <article>, fallback to <body>.
    - Remove scripts, styles, nav, header/footer, forms, asides.
    - Join paragraphs, collapse whitespace, normalize accents.
    """
    try:
        soup = BeautifulSoup(html or "", "lxml")
        container = soup.find('main') or soup.find('article') or
soup.body or soup
        for tag in
container.find_all(['script','style','nav','footer','header','form','a
side']):
            tag.decompose()
        paragraphs = [p.get_text(" ", strip=True) for p in
container.find_all('p')]
        text = " ".join(paragraphs) if paragraphs else
container.get_text(" ", strip=True)
        text = re.sub(r'\s+', ' ', text).strip()
        return unidecode(text)
    except Exception:
        return ""

def extract_title(html: str) -> str:
    """
    Extract title using <title>, then og:title, then <h1> as
fallbacks.
    """
    try:
        soup = BeautifulSoup(html or "", "lxml")
        if soup.title and soup.title.string:
            return soup.title.string.strip()
        og = soup.find("meta", property="og:title")
        if og and og.get("content"):
            return og.get("content").strip()
        h1 = soup.find("h1")
        if h1:
            return h1.get_text(" ", strip=True)
        return ""
    except Exception:
        return ""

# Language detection to support multilingual pipelines (simple
heuristic)
def detect_language(text: str) -> str:
    """
    Minimal/fast language hint; replace with langid/fasttext in
production if needed.
    """
    # Heuristic: if many non-ASCII, mark 'non-en'; better: langid or
fasttext lid.176
    try:
        ascii_ratio = sum(ord(c) < 128 for c in text) / max(1,
len(text))
        return "en" if ascii_ratio > 0.95 else "non-en"
    except Exception:
        return "unknown"

# Parse rows
parsed = []
for _, row in df.iterrows():
    html = row.get('html_content') or ''
    title = extract_title(html)
    body = extract_main_text(html)
    wc = len(body.split())
    lang = detect_language(body)
    parsed.append({"url": row['url'], "title": title, "body_text":
body, "word_count": wc, "lang": lang})

parsed_df = pd.DataFrame(parsed)

# Save extracted content CSV required by deliverables
parsed_df.to_csv(f"{base}/data/extracted_content.csv", index=False)
parsed_df.head(3)
```

```
{"summary":"{\n  \"name\": \"parsed_df\",\n  \"rows\": 81,\n
\"fields\": [\n    {\n      \"column\": \"url\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 81,\n        \"samples\": [\n
\"https://comfax.com/reviews/free-fax/\",\n          \"https://www.cm-
alliance.com/cybersecurity-blog\",\n
\"https://en.wikipedia.org/wiki/Remote_desktop_software\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"title\",\n      \"properties\":
```

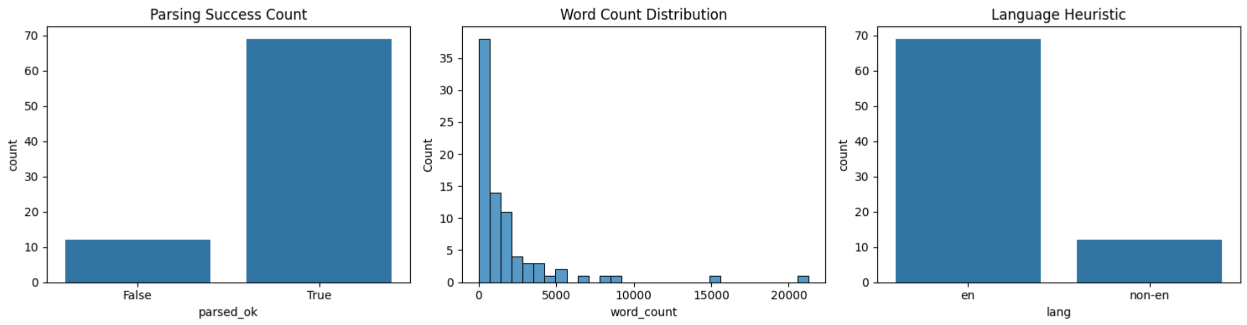
```
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 70,\
n        \"samples\": [\n          \"Zero Trust Network Access (ZTNA)
\\u2013 Benefits & Overview | Zscaler\",\n          \"Cyber Security
Blog\",\n          \"The Original $1 Dollar Makeup, Cosmetics and
Beauty Online Shop \\u2013 Shop Miss A\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"body_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 70,\n        \"samples\": [\n          \"/ What
Is Zero Trust Network Access (ZTNA)? Zero trust network access (ZTNA)
is a set of technologies that enable secure remote access to internal
applications. Trust is never granted implicitly, and access is granted
on a need-to-know, least-privileged basis defined by granular
policies. ZTNA gives users secure connectivity to private apps without
placing them on the network or exposing apps to the internet. The
future of work is distributed, and remote workforces and cloud
workloads demand secure remote access. However, traditional remote
access solutions like virtual private networks (VPNs) are not flexible
or granular enough for distributed environments, increasing breach
risks. This is a key reason 65% of enterprises recently reported plans
to replace their VPNs with a solution such as ZTNA. ZTNA provides
secure remote access to internal applications for any user, from
anywhere, without putting critical resources at risk. To accomplish
this, it starts with an architecture that's fundamentally different
from a network-centric solution. Relying on a software-defined
perimeter (SDP) , ZTNA enforces secure, identity-based access
controls. This helps organizations replace their VPNs while reducing
dependence on tools like DDoS protection, global load balancing, and
firewalls. Following four core principles, ZTNA: VPN VPNs give users
access to a network and its resources through an encrypted, private
tunnel. For many years, they were sufficient for users who worked
remotely on occasion. However, cloud and remote work trends in the
mid-to-late 2010s began to highlight shortcomings in the VPN approach.
ZTNA ZTNA provides secure least-privileged access. Instead of granting
trust based only on credentials, it verifies users based on a breadth
of context, including device, location, and identity, for every access
request. Once verified, users receive direct application access rather
than network-wide access. With VPNs creating serious compliance and
security risks, more and more organizations are discovering the
advantages of ZTNA. Here are some of the top reasons to make the
switch: ZTNA helps organizations strengthen their overall security
posture and agility by delivering: While ZTNA has many use cases, most
organizations start with one of these four. VPN Alternative VPNs are
inconvenient and slow for users, difficult to manage, and offer poor
security. More than half of organizations cite security and poor user
experiences as the top challenges of VPN solutions. Reduce Third-Party
Risk Most third-party users receive overprivileged access, and they
largely use unmanaged devices, both of which introduce risks. ZTNA
significantly reduces third-party risk by never providing direct
```

```
network access and enforcing least-privileged access to apps.
Accelerate M&A Integration M&A integrations can span multiple years as
organizations converge networks and deal with overlapping IPs. ZTNA
can provide direct app access with no need to converge networks or
resolve IP overlap, significantly simplifying and speeding up M&A
value capture. Secure Multicloud Access Securing hybrid and multicloud
access is the most popular place for organizations to start their ZTNA
journey. As more companies adopt the cloud, the vast majority are
turning to ZTNA for security and access control for their multicloud
strategies. How Does ZTNA Simplify Multicloud Access? ZTNA simplifies
multicloud access by providing secure, direct connections between
users and specific apps, wherever they are. It eliminates the need for
complex network-level configurations or redundant VPNs, using
identity-based authentication and granular access controls to unify
security across clouds. Implementing ZTNA follows a phased approach
designed to ensure smooth adoption, enhance security, and reduce
risks: In today's crowded marketplace, it's important to consider
several other key criteria when evaluating ZTNA solutions against your
unique needs: Keep these things in mind as you look for the vendor
that complements your goals and vision . Zscaler Private Access(tm) is
the world's most deployed ZTNA platform, built on the unique Zscaler
zero trust architecture. As a cloud native service, ZPA can be
deployed in hours to replace legacy VPNs and remote access tools with
a holistic zero trust platform. Zscaler Private Access delivers: Is
ZTNA More Secure Than VPN? ZTNA is more secure than VPNs because it
gives access only to specific apps instead of entire networks. This
reduces risks like lateral movement, hiding sensitive systems from
attackers, and shrinking the attack surface for better protection.
Which Industries Need ZTNA the Most? Industries like healthcare,
finance, and tech may gain the most from ZTNA. However, for any
organization that depends on remote teams, strict rules, or large
networks, ZTNA helps them keep data and apps safe with least-
privileged access. How Hard Is ZTNA to Deploy and Manage? ZTNA is
simple to set up and oversee. It works with cloud-based systems, so it
deploys in days, not weeks. Its portals offer quick control of
policies, instant user insights, and easy scaling for growth. Can ZTNA
Improve Cybersecurity in Hybrid Work Environments? ZTNA boosts
security for hybrid work by limiting access to apps, stopping lateral
movement, and changing policies based on device and location. It
guards systems without slowing down or complicating user access. Is
ZTNA a Good Alternative to Legacy Network Segmentation? ZTNA is ideal
for replacing network segmentation. It uses identity-based app
permissions instead of complex network setups, removing over-access
risks while simplifying security for workflows and cloud setups. SD-
WAN vs. MPLS: What's the Difference? What Is Trusted Internet
Connections (TIC) 3.0? What's the Difference Between SDP and VPN?\",\n
\"Cybersecurity Blog Cybersecurity Measures Every Financial Business
Should Take In the modern digital world, financial institutions are
exposed to an unmatched wave of cyber attacks. Financial organisations
```

```
are now on the front line against cyber attacks with... 31 October
2025 A Complete Guide to a Career in Cyber Security in 2025 More and
more companies are being hit by cyber attacks, which means they need
skilled people to keep their data safe. Because of this, cyber
security has become one of the most... 31 October 2025 Game Over: 5
Types of Cyber Attacks Threatening the Gaming World Online gaming has
massively exploded in popularity in the last decade. From fast-paced
shooters to multiplayer games like Valorant, players from around the
world log in daily to... 30 October 2025 Faster Incident Response: Top
Tech Tricks for Cybersecurity Experts When the stakes are high, time
is the enemy. The longer it takes to detect, assess, and contain a
cyber incident, the higher the damage and cost. Today's
cybersecurity... 30 October 2025 How Data-Driven Insights Are Shaping
the Future of Cyber Defence In a world where cyber attacks are
becoming more sophisticated and dangerous, traditional security
measures like firewalls are no longer enough. Hackers are using
advanced tools... 29 October 2025 Mobile Proxies: The Hidden Tool
Behind Smarter Digital Workflows Let's face it -- working online today
isn't as simple as it used to be. Websites block you after too many
logins, social platforms ban accounts that look \\\"suspicious,\\\"
and search... 28 October 2025 How to Ensure Maximum Security in Your
Data Room Virtual data rooms (VDRs) are already indispensable in the
complex finance world. Business owners and dealmakers use them to
manage sensitive information during such high-stakes... 28 October
2025 Best Proxy Providers for Cybersecurity Threat Intelligence
Cybersecurity threat intelligence relies on data. Massive amounts of
it. But gathering that data safely and anonymously is not simple.
Analysts need to monitor malicious domains,... 28 October 2025 Simple
Cybersecurity Tips for Every Online Business For every online
business, regardless of the industry, there is something arguably more
important than boosting sales, and that's taking care of
cybersecurity. With countless... 27 October 2025 Why Choose a Fintech-
First Banking Platform Over Traditional Banks? Running a business with
traditional banks can feel slow and costly. Studies show fintech
platforms are often faster and more affordable for companies. This
blog will guide you... 27 October 2025 10 Emerging Cyber Threats In
2026: How To Spot Them & Mitigate Risks Emerging cyber threats are
racing through blind spots that didn't even exist a year ago. What
used to be a \\\"rare exploit\\\" is now a Tuesday afternoon. And the
problem is that most... 27 October 2025 The Importance of Password
Security for Students in the Digital Age More and more, students are
using the internet to learn, talk to each other, and stay organized.
Students use a lot of online services all the time, such as email,
social media,... 23 October 2025 The Hidden Digital Risks in New
Business Acquisitions Stepping into new business ownership can feel
exhilarating yet daunting. For instance, when you see an auto shop for
sale, the potential for growth and profit is immense, but the... 23
October 2025 Can Students Lose Their Data When Using AI Tools
Artificial Intelligence (AI) is changing the way students learn,
study, and complete their assignments. Whether it's using ChatGPT to
understand a tricky topic or relying on AI... 16 October 2025 10 Cyber
Resilience Exercises That Go Beyond Penetration Testing Organizations
often focus their cybersecurity programs on penetration testing to
uncover system weaknesses before attackers do. While pen testing
remains valuable, it offers only... 15 October 2025 Why Cybersecurity
Training Is a Must for Students Imagine losing your class notes,
assignment files, or even your entire student loan account - just
because you clicked one bad link or reused an old password. Scary,
right? In... 14 October 2025 How Leading Companies Are Redefining
Secure Remote Work In 2025 Remote work has changed how businesses
operate, but it also brings new security risks. In 2025, leading
companies will adopt sophisticated tools and strategies to protect
digital... 3 October 2025 Tabletop to Keyboard: Baking VDI Realities
into Incident Response When an incident hits an estate built on End-
User Computing (EUC) and hosted desktops, the neat diagrams in many
playbooks develop hairline cracks. Decisions that seem obvious on... 3
October 2025 Sept 2025: Biggest Cyber Attacks, Ransomware Attacks and
Data Breaches If September 2025 proved anything, it's that no industry
is safe from the relentless wave of cyber crime. In just one month, we
witnessed attacks on Jaguar Land Rover (JLR) and... 1 October 2025
Designing an Executable DDoS Runbook: First Packet to Full Recovery
Two truths surface the moment a DDoS hits: time compresses, and
options shrink. A runbook that lives only in a wiki will not help your
team make the next correct move under... 30 September 2025 Cyber
Tabletop Exercises 2025: Top Tips for an Effective Cyber Drill If
you're caught up on world news, you couldn't have missed the recent
cyber attacks on Marks and Spencer, Jaguar Land Rover and the European
Airports. These attacks have yet... 26 September 2025 Dedicated vs
Shared Datacenter Proxies: How to Choose the Best Option A reliable
datacenter proxy server provides an array of useful benefits for both
personal and professional use. You can use it to improve your online
security, protect your... 25 September 2025 What Is Cyber Threat
Hunting? Definition, Examples and Useful Tools Cyber attackers succeed
because they don't always set off alarms, and automated systems only
catch what they already know to look for. That gap is where cyber
threat hunting... 23 September 2025 How to Protect Your Crypto Assets
from Digital Theft Although there are fast gains and borderless
finance in crypto trading, investors must also deal with the risk of
digital theft. Whether it's a phishing scam, a hacked exchange,... 23
September 2025 European Airports Cyber Attack: ENISA Confirms Third-
Party Ransomware The European aviation sector faced a stark reminder
of its digital fragility over the weekend (of September 19- 21, 2025).
A \\\"cyber-related\\\" disruption crippled passenger... 22 September
2025 Leveraging AI Tools to Enhance Your Technical Writing Imagine
this: it is late at night, 2 AM, you have a 10-page essay to finish
overnight, and when you look at the screen of your laptop, all you see
are six words: How to Secure... 19 September 2025 From Prototype to
Market Leader: Mapping the Web Development Lifecycle In 2025,
launching a web platform is no longer a task confined to coding or
visual design - it is a comprehensive, strategic process. Leading
businesses treat web development as a... 15 September 2025 Anatomy of
a BEC Attack: What AI Can Detect That Humans Might Miss The upcoming
challenge with email security today is that attackers are increasingly
leaning into social engineering scams, rather than the malware-ridden
emails we all know and... 15 September 2025 Cybersecurity Recruitment
Guide for HR: 2025 Best Practices Cybersecurity has shifted from being
a specialised IT function to a core business priority. With rising
data breaches, ransomware attacks, and regulatory pressures, every...
11 September 2025 Cyber Incident Response Playbook Examples for 2025
Have you read about the massive Salesloft-Drift breach? Did you follow
how the Marks and Spencer cyber attack brought the iconic retail brand
to its knees? Recently luxury... 11 September 2025 Asset Inventory
Management Software Explained Businesses need to manage assets
efficiently. Properly tracking equipment, tools, and resources enables
companies to operate smoothly, which is where asset inventory
management... 10 September 2025 Salesloft-Drift Attack: One
Compromised Integration Shakes 700+ Cos This month, the world has
witnessed one of the largest ever SaaS supply-chain breaches in
history. Yes, we are talking about the Salesloft-Drift breach. 9
September 2025 Phishing 3.0: AI and Deepfake-Driven Social Engineering
Attacks Phishing is no longer an easy-to-detect cyber attack. With the
rise of artificial intelligence, attackers now launch AI-driven
phishing campaigns to mimic human behaviour. They... 8 September 2025
JLR Cyber Incident Marks Latest Blow in UK's Cyber Crime Wave Jaguar
Land Rover (JLR) disclosed a \\\"cyber incident\\\" that forced the
luxury carmaker to proactively shut down systems, severely disrupting
manufacturing and global retail... 4 September 2025 The Hidden Costs
of Running Microsoft Access Without Support In today's fast-paced
digital landscape, businesses rely heavily on data management systems
to handle their crucial operations. Microsoft Access, a trusted and
widely used... 3 September 2025 Why Scalable Annotation Platforms Are
Key to Enterprise AI Enterprise AI needs more than just models. It
runs on labelled data, at scale, across teams, and often under tight
deadlines. A one-size-fits-all tool won't work here. You need a... 1
September 2025 Major Cyber Attacks, Ransomware Attacks and Data
Breaches: August 2025 What does an insurance giant, a luxury fashion
house, an airline, and even Google's own Salesforce platform have in
common? Unfortunately, in August 2025, the answer was hackers.... 1
September 2025 Top Personal Cybersecurity Measures To Take when
Trading in Crypto Starting as a niche experiment, cryptocurrencies
have now become a mainstream asset class. This growth hasn't been all
smooth sailing for investors. Crypto is an entirely new... 29 August
2025 Mitigate Fraud Risk with Real-Time Business Registration
Verification Business fraud has evolved from simple deception into
sophisticated digital warfare. Companies now face threats from fake
vendors, shell companies, and imposter businesses that... 29 August
2025 How to Travel Light and Secure: Cybersecurity Tips for Japan in
```

```
2025 Japan attracts millions of visitors each year thanks to its
intriguing balance between tradition and innovation, but
unfortunately, along with the convenience of digital... 29 August 2025
Outsourcing MVP Development: Pros, Cons, and Best Practices Every
founder with a big idea knows the surge of excitement that comes with
moving from concept to the possibility of creating something real that
your users can experience at... 28 August 2025 Five Red Flags in Your
Email Security Strategy While email is a critical means of
communication, it is also a major risk area. Poor email security
exposes the organization to data breaches, financial losses, and
damage to... 28 August 2025 11 Top Features in AML Compliance Software
for Modern Security Teams Modern AML compliance software should lower
false positives, improve the speed it takes to resolve cases, and
provide audit-ready evidence on demand. Choosing the right AML... 27
August 2025 The Rise Of Edge Computing In IoT App Development The
Internet of Things (IoT) has revolutionised how we interact with the
world, embedding intelligence into everyday objects from smart
thermostats to industrial sensors. As... 26 August 2025 Cyber Security
Drill Examples: Top Tabletop Exercise Scenarios in 2025 In 2025, cyber
attacks occur every 39 seconds on average as per some estimates. A
projected 160 reports of daily attacks emerge in the US alone. The
frequency and sophistication... 26 August 2025 Cyber Incident Response
Playbook: An Imperative for Businesses in 2025 In 2025, it's no longer
a question of if your organisation will face a cyber attack--it's
when. And when that moment comes, will your team know exactly what to
do? The pervasive... 21 August 2025 Top WebRTC App Development
Companies in 2025 The WebRTC development landscape has evolved
dramatically, presenting businesses with two distinct paths: quick
deployment through Communication Platform as a Service (CPaaS)... 19
August 2025 Strengthening Business Communication Security with VoIP
Technology For any business that wants to be successful in today's
dynamic landscape, effective communication is key. As a matter of
fact, it wouldn't be wrong to state that effective... 18 August 2025
Cracking the Coinbase Breach: What Went Wrong and What We Can Learn
One of the most influential Cryptocurrency platforms in the world,
Coinbase, is the latest victim of a headline-making cyber attack.
While there have been other crypto-related... 15 August 2025 7 Tactics
to Stop Deepfake Attacks from Deceiving Your Executive Team In May
2024, scammers set up a bogus Microsoft Teams meeting, used an AI-
cloned voice and YouTube footage of WPP's CEO Mark Read, and attempted
to convince an agency leader they... 14 August 2025 Simply fill in
your details to request a FREE callback\",\n         \"4.6 / 5.0 4.85
/ 5.0 5.0 / 5.0 5.0 / 5.0 4.2 / 5.0 4.75 / 5.0 4.75 / 5.0 4.59 / 5.0
4.59 / 5.0 4.71 / 5.0 4.21 / 5.0 4.64 / 5.0 4.53 / 5.0 4.73 / 5.0 4.34
/ 5.0 4.66 / 5.0 4.79 / 5.0 4.12 / 5.0 4.71 / 5.0 And optional subtext
4.61 / 5.0 4.42 / 5.0 4.08 / 5.0 4.38 / 5.0 4.67 / 5.0 4.48 / 5.0 4.47
/ 5.0 4.76 / 5.0 3.71 / 5.0 3.84 / 5.0 4.23 / 5.0 3.79 / 5.0 4.44 /
5.0 3.93 / 5.0 3.92 / 5.0 4.57 / 5.0 5.0 / 5.0 And optional subtext
5.0 / 5.0 5.0 / 5.0 4.5 / 5.0 5.0 / 5.0 5.0 / 5.0 5.0 / 5.0 And
```

```
optional subtext 4.19 / 5.0 4.44 / 5.0 4.86 / 5.0 4.9 / 5.0 4.69 / 5.0
4.52 / 5.0 4.66 / 5.0 4.74 / 5.0 4.59 / 5.0 4.76 / 5.0 4.46 / 5.0 4.29
/ 5.0 4.63 / 5.0 4.91 / 5.0 4.45 / 5.0 4.83 / 5.0 4.35 / 5.0 4.5 / 5.0
4.67 / 5.0 4.59 / 5.0 4.51 / 5.0 3.65 / 5.0 4.69 / 5.0 3.64 / 5.0 4.63
/ 5.0 4.69 / 5.0 4.52 / 5.0 4.34 / 5.0 4.29 / 5.0 4.6 / 5.0\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"word_count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3240,\n        \"min\": 0,\n        \"max\": 21292,\n
\"num_unique_values\": 67,\n        \"samples\": [\n          3186,\n
197,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"lang\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"non-
en\",\n          \"en\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"parsed_df"}

# Parsing diagnostics visuals
import matplotlib.pyplot as plt
import seaborn as sns

parsed_df['parsed_ok'] = parsed_df['body_text'].str.len() > 0

fig, ax = plt.subplots(1, 3, figsize=(15,4))
sns.countplot(data=parsed_df, x='parsed_ok', ax=ax[0])
ax[0].set_title("Parsing Success Count")

sns.histplot(parsed_df['word_count'], bins=30, ax=ax[1])
ax[1].set_title("Word Count Distribution")

sns.countplot(data=parsed_df, x='lang', ax=ax[2])
ax[2].set_title("Language Heuristic")
plt.tight_layout()
plt.show()
```



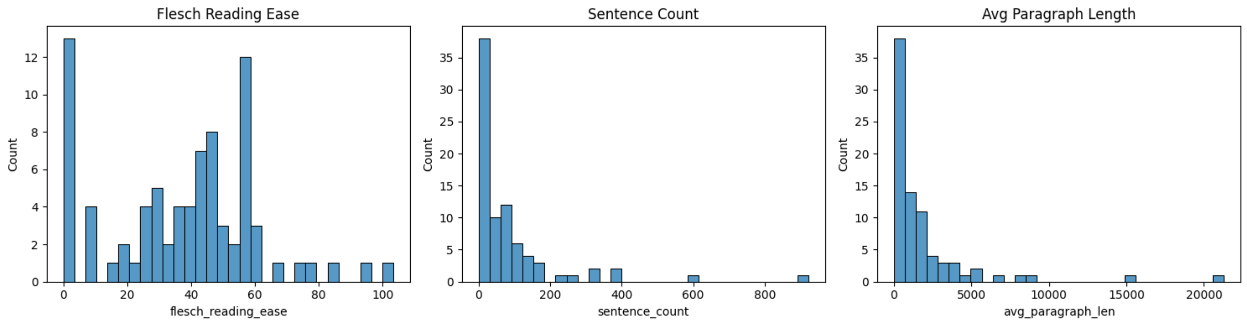
Preprocessing & feature engineering (readability, TF-IDF, embeddings), visuals; structural features

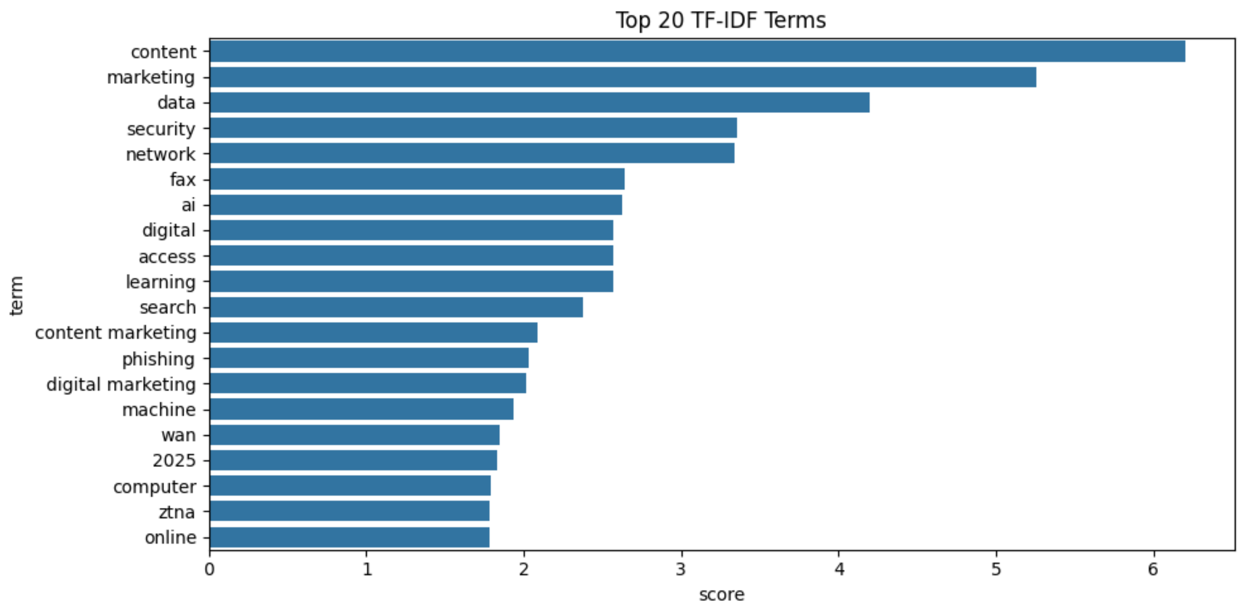
```python
import numpy as np
import textstat
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

# Basic cleaners and metrics
def clean_text_basic(s: str) -> str:
    """
    Lowercase, strip, collapse whitespace; light normalization for
embeddings/TF-IDF.
    """
    s = (s or "").lower().strip()
    s = re.sub(r'\s+', ' ', s)
    return s

def sentence_count(s: str) -> int:
    """
    Approximate sentence count using NLTK; try/except for robustness
on edge cases.
    """
    try:
        return len(sent_tokenize(s)) if s else 0
    except Exception:
        return 0

def flesch_reading_ease_score(s: str) -> float:
    """
    Flesch Reading Ease:
    206.835 − 1.015 × (words/sentences) − 84.6 × (syllables/words)
    Note: use advisory only; pair with other UX features.
    """
    try:
        return float(textstat.flesch_reading_ease(s)) if s and
len(s.split()) >= 5 else 0.0
    except Exception:
        return 0.0

# Structural/UX features to complement readability
def approx_avg_paragraph_len(s: str) -> float:
    """
    Approximate paragraph length using double-newline or fallback to 1
paragraph.
    """
    paras = re.split(r'\n\s*\n', s) if s else []
    paras = [p.strip() for p in paras if p.strip()]
    if not paras:
        return float(len(s.split()))
    lens = [len(p.split()) for p in paras]
    return float(np.mean(lens))

def list_density(s: str) -> float:
    """
    Heuristic list density: count of bullet-like tokens versus
sentence_count.
    """
    bullets = len(re.findall(r'(^-|\n-|\* |\d+\.)', s))
    sc = max(1, sentence_count(s))
    return float(bullets) / sc

feat_df = parsed_df.copy()
feat_df['clean_text'] = feat_df['body_text'].apply(clean_text_basic)
feat_df['sentence_count'] = feat_df['body_text'].apply(sentence_count)
feat_df['flesch_reading_ease'] =
feat_df['body_text'].apply(flesch_reading_ease_score)
feat_df['avg_paragraph_len'] =
feat_df['body_text'].apply(approx_avg_paragraph_len)
feat_df['list_density'] = feat_df['body_text'].apply(list_density)

# TF-IDF for keywords and vectors
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2),
stop_words='english')
tfidf_matrix = tfidf.fit_transform(feat_df['clean_text'])
feature_names = np.array(tfidf.get_feature_names_out())

# Extract top-5 keywords per doc
top_keywords = []
for row in range(tfidf_matrix.shape[0]):
    vec = tfidf_matrix.getrow(row)
    if vec.nnz == 0:
        top_keywords.append("")
        continue
    top_idx = np.argsort(vec.toarray()[0])[-5:][::-1]
    kws = feature_names[top_idx]
    top_keywords.append("|".join(kws))
feat_df['top_keywords'] = top_keywords

# Sentence-transformer embeddings for semantic similarity
from sentence_transformers import SentenceTransformer
set_seeds(42)
embedder = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = embedder.encode(feat_df['clean_text'].tolist(),
convert_to_numpy=True, normalize_embeddings=True)

# Assemble features CSV
feat_out =
feat_df[['url','word_count','sentence_count','flesch_reading_ease','av
g_paragraph_len','list_density','top_keywords']].copy()
```

```
feat_out['embedding'] = [emb.tolist() for emb in embeddings]
feat_out.to_csv(f"{base}/data/features.csv", index=False)
feat_out.head(3)
```

```
{"summary":"{\n  \"name\": \"feat_out\",\n  \"rows\": 81,\n
\"fields\": [\n    {\n      \"column\": \"url\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 81,\n        \"samples\": [\n
\"https://comfax.com/reviews/free-fax/\",\n          \"https://www.cm-
alliance.com/cybersecurity-blog\",\n
\"https://en.wikipedia.org/wiki/Remote_desktop_software\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"word_count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3240,\n        \"min\": 0,\n        \"max\": 21292,\n
\"num_unique_values\": 67,\n        \"samples\": [\n          3186,\n
197,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sentence_count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 140,\n        \"min\": 0,\n
\"max\": 923,\n        \"num_unique_values\": 56,\n
\"samples\": [\n          60,\n          88,\n          3\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"flesch_reading_ease\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
23.81769247561053,\n        \"min\": 0.0,\n        \"max\":
103.39199641553986,\n        \"num_unique_values\": 69,\n
\"samples\": [\n          41.70478781209161,\n
41.55896708161657,\n          60.428000000000026\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"avg_paragraph_len\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3240.8523253541016,\n        \"min\": 0.0,\n        \"max\": 21292.0,\
n        \"num_unique_values\": 67,\n        \"samples\": [\n
3186.0,\n          197.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"list_density\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15.997460271465892,\n        \"min\": 0.0,\n        \"max\": 144.0,\n
\"num_unique_values\": 33,\n        \"samples\": [\n
0.028169014084507043,\n          0.015037593984962405,\n
0.31313131313131315\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"top_keywords\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 70,\n        \"samples\":
[\n          \"ztna|access|vpns|network|secure\",\n          \"2025|
cyber|september 2025|september|october 2025\",\n          \"59|69|71|
67|66\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"embedding\",\n      \"properties\": {\n        \"dtype\":
```

```
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"feat_out"}

# Feature distribution
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots(1, 3, figsize=(15,4))
sns.histplot(feat_df['flesch_reading_ease'], bins=30, ax=ax[0])
ax[0].set_title("Flesch Reading Ease")

sns.histplot(feat_df['sentence_count'], bins=30, ax=ax[1])
ax[1].set_title("Sentence Count")

sns.histplot(feat_df['avg_paragraph_len'], bins=30, ax=ax[2])
ax[2].set_title("Avg Paragraph Length")
plt.tight_layout()
plt.show()

# Top TF-IDF terms
term_scores = np.asarray(tfidf_matrix.sum(axis=0)).ravel()
term_df = pd.DataFrame({"term": feature_names, "score":
term_scores}).sort_values("score", ascending=False).head(20)

plt.figure(figsize=(10,5))
sns.barplot(data=term_df, x="score", y="term", orient="h")
plt.title("Top 20 TF-IDF Terms")
plt.tight_layout()
plt.show()
```



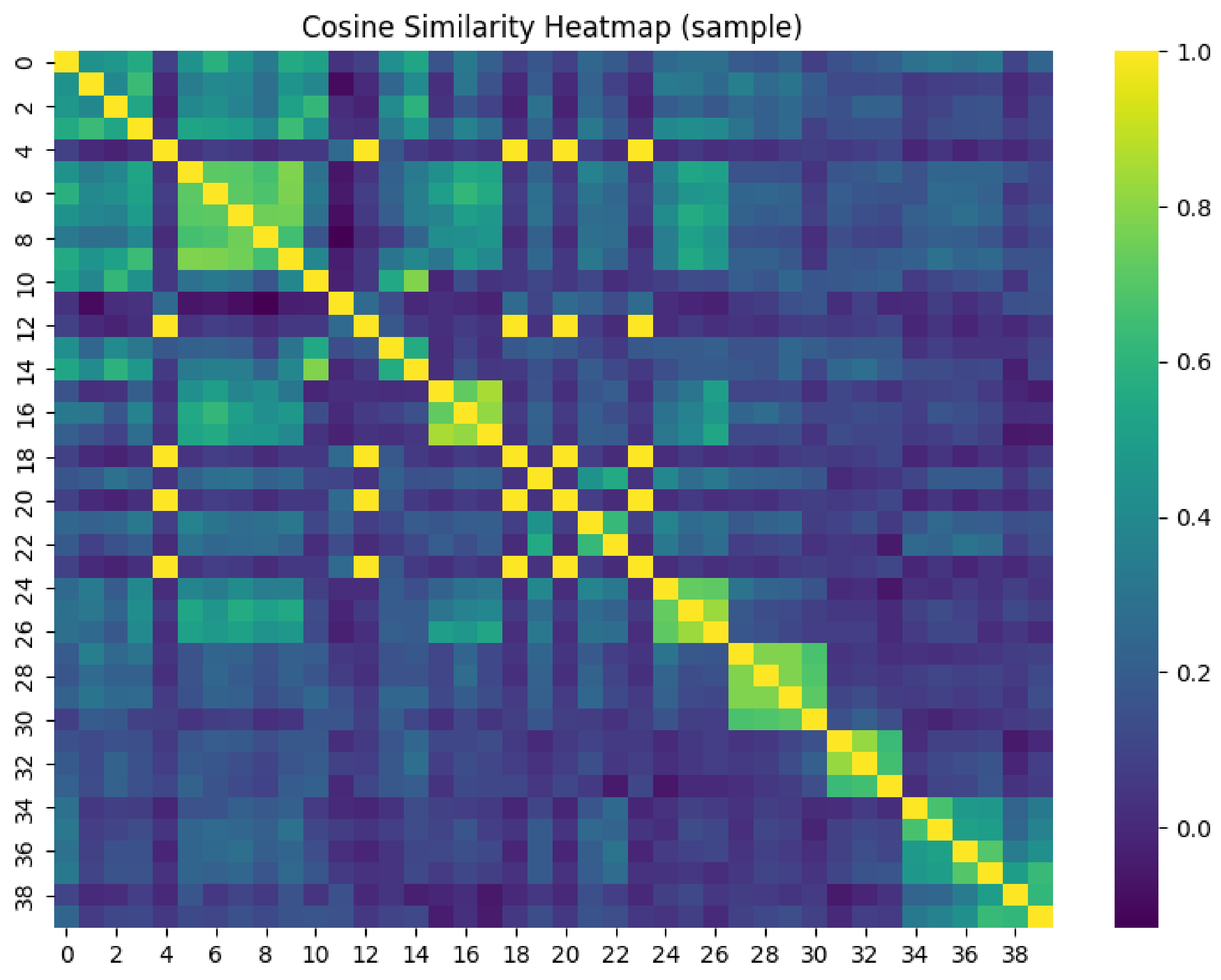


Duplicate detection: MinHash LSH candidates + cosine rerank; thin content; heatmap

```
from datasketch import MinHash, MinHashLSH
from sklearn.metrics.pairwise import cosine_similarity

# Helper to shingle text and build MinHash signatures
def get_shingles(text: str, k: int = 5):
    """
    Create word shingles of size k for Jaccard similarity; lowercase
first.
    """
    toks = (text or "").lower().split()
    return {" ".join(toks[i:i+k]) for i in range(max(0, len(toks)-
k+1))}

def minhash_signature(shingles, num_perm=128):
    """
    Generate a MinHash signature for a set of shingles.
    """
    m = MinHash(num_perm=num_perm)
    for s in shingles:
        m.update(s.encode('utf8'))
    return m

# Build LSH index for lexical candidate generation (sublinear)
num_perm = 128
lsh_threshold = 0.7  # lexical Jaccard candidate threshold; to be
tuned
lsh = MinHashLSH(threshold=lsh_threshold, num_perm=num_perm)
signatures = []
```

```
for i, text in enumerate(feat_df['clean_text']):
    sh = get_shingles(text, k=5)
    sig = minhash_signature(sh, num_perm=num_perm)
    lsh.insert(f"doc-{i}", sig)
    signatures.append(sig)

# Re-rank with cosine similarity over embeddings to finalize
duplicates
SIM_THRESHOLD = 0.80  # semantic cosine threshold; tune on a small
labeled set
pairs = []
for i in range(len(feat_df)):
    # Retrieve lexical candidates quickly
    cands = lsh.query(signatures[i])
    # Convert keys back to indices
    cand_idx = [int(c.split("-")[1]) for c in cands if
int(c.split("-")[1]) > i]
    if not cand_idx:
        continue
    # Compute cosine similarities only for candidates
    sims = cosine_similarity(embeddings[i].reshape(1, -1),
embeddings[cand_idx]).ravel()
    for idx, s in zip(cand_idx, sims):
        if s >= SIM_THRESHOLD:
            pairs.append({
                "url1": feat_df['url'].iloc[i],
                "url2": feat_df['url'].iloc[idx],
                "similarity": float(s)
            })

dup_df = pd.DataFrame(pairs)
dup_df.to_csv(f"{base}/data/duplicates.csv", index=False)

# Thin content flag
THIN_THRESHOLD = 500
feat_view = feat_out.copy()
feat_view['is_thin'] = feat_df['word_count'] < THIN_THRESHOLD

# Summary metrics
summary = {
    "total_pages": int(len(feat_df)),
    "duplicate_pairs": int(len(dup_df)),
    "thin_pages": int(feat_view['is_thin'].sum()),
    "thin_pct": round(100.0 * feat_view['is_thin'].mean(), 2)
}
summary
```

```
{'total_pages': 81,
 'duplicate_pairs': 66,
 'thin_pages': 35,
 'thin_pct': np.float64(43.21)}

# Similarity heatmap visual

# Compute a small subset pairwise matrix for readability
idx = np.arange(min(40, len(feat_df)))
sim_subset = cosine_similarity(embeddings[idx], embeddings[idx])

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.heatmap(sim_subset, cmap="viridis")
plt.title("Cosine Similarity Heatmap (sample)")
plt.tight_layout()
plt.show()
```



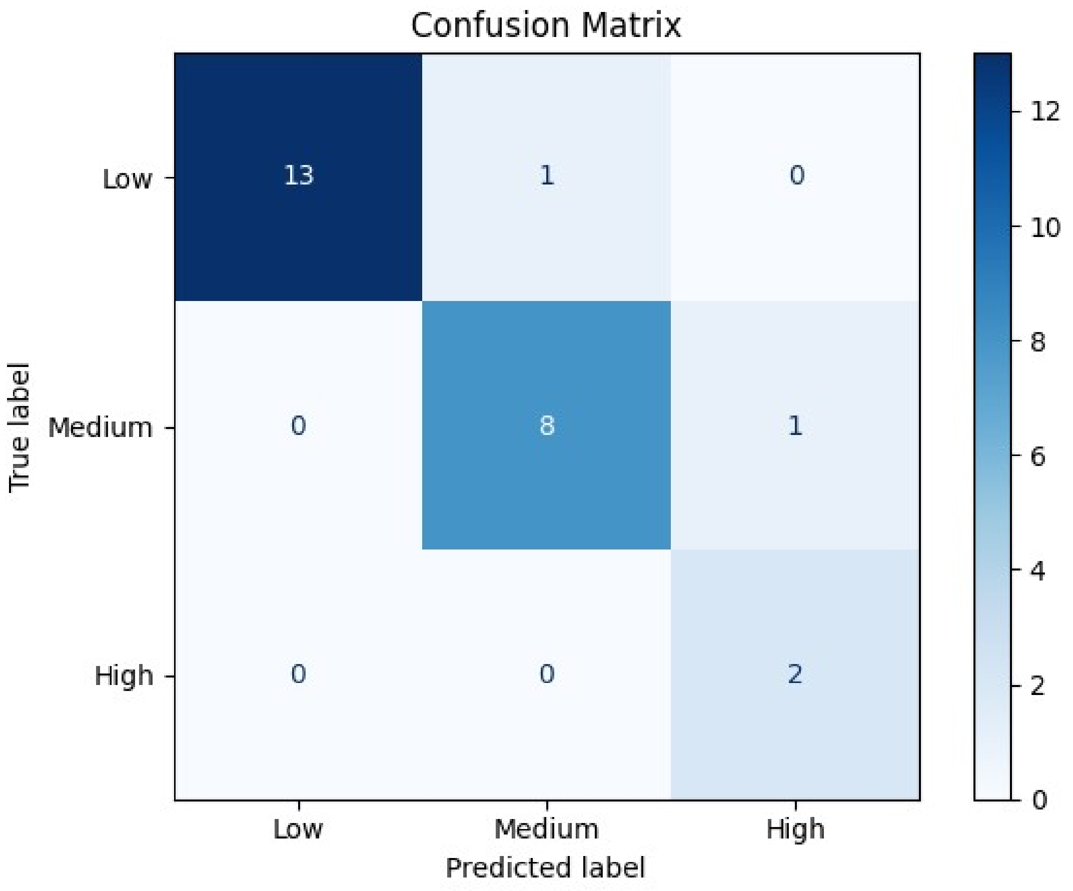
Labels, model, baseline, metrics, confusion matrix, feature importance

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,
f1_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
import joblib

# Label rules per spec
def label_quality(row):
    wc = row['word_count']; fr = row['flesch_reading_ease']
    if (wc > 1500) and (50 <= fr <= 70): return "High"
    if (wc < 500) or (fr < 30): return "Low"
    return "Medium"

lab_df = feat_view.copy()
lab_df['quality_label'] = lab_df.apply(label_quality, axis=1)

# Feature set (structural features)
X =
lab_df[['word_count','sentence_count','flesch_reading_ease','avg_parag
raph_len','list_density']]
y = lab_df['quality_label']

# Split with stratification
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)

# Baseline: word_count only rule
def baseline_predict(row):
    return "High" if row['word_count'] > 1500 else ("Low" if
row['word_count'] < 500 else "Medium")

y_pred_base = X_test.apply(baseline_predict, axis=1)
baseline_acc = accuracy_score(y_test, y_pred_base)
baseline_f1 = f1_score(y_test, y_pred_base, average='weighted')

# Model: RandomForest (robust + interpretable)
set_seeds(42)
clf = RandomForestClassifier(
    n_estimators=300,
    random_state=42,
    class_weight="balanced_subsample",
    max_depth=None
)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

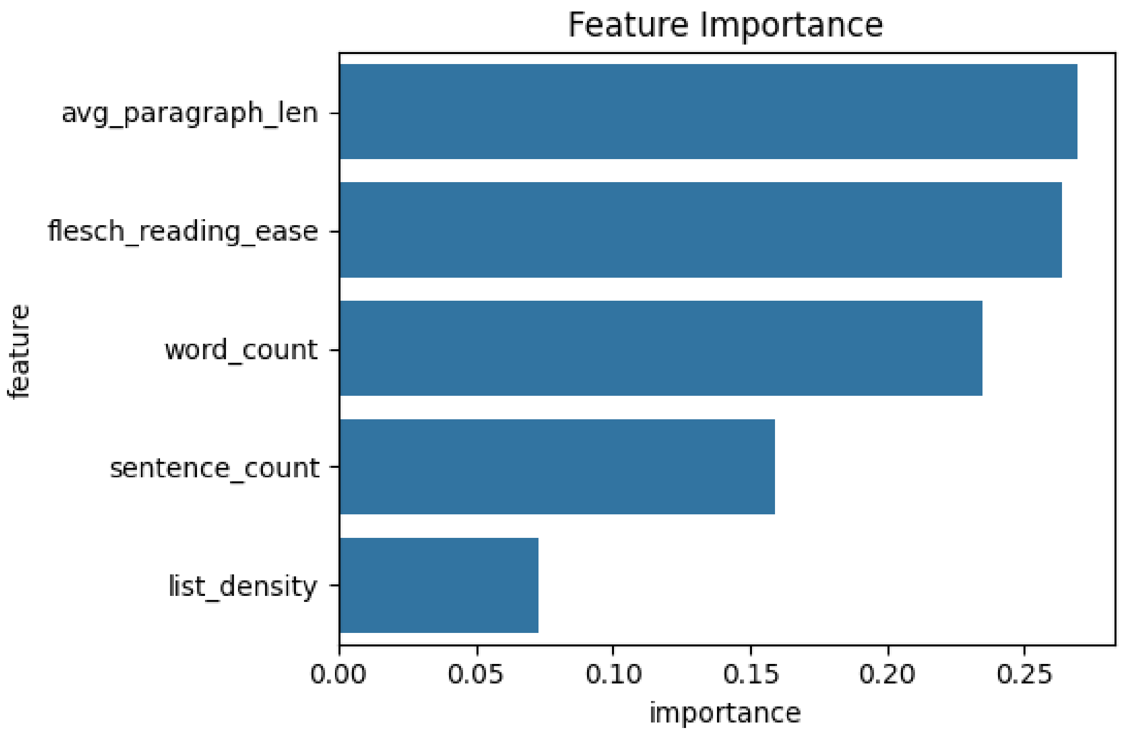
```
acc = accuracy_score(y_test, y_pred)
f1w = f1_score(y_test, y_pred, average='weighted')
print("Model Performance:\n", classification_report(y_test, y_pred,
digits=2))
print(f"Overall Accuracy: {acc:.3f} | Baseline Accuracy:
{baseline_acc:.3f}")
print(f"Weighted F1: {f1w:.3f} | Baseline Weighted F1:
{baseline_f1:.3f}")

# Persist model
joblib.dump(clf, f"{base}/models/quality_model.pkl")
```

```
Model Performance:
               precision    recall  f1-score   support

        High       0.67      1.00      0.80         2
         Low       1.00      0.93      0.96        14
      Medium       0.89      0.89      0.89         9

    accuracy                           0.92        25
   macro avg       0.85      0.94      0.88        25
weighted avg       0.93      0.92      0.92        25

Overall Accuracy: 0.920 | Baseline Accuracy: 0.600
Weighted F1: 0.923 | Baseline Weighted F1: 0.648

['seo-content-detector/models/quality_model.pkl']
```

```
# Confusion matrix & importance
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred, labels=["Low","Medium","High"])
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=["Low","Medium","High"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix")
plt.tight_layout()
plt.show()

importances = clf.feature_importances_
fi = pd.DataFrame({"feature": X.columns, "importance":
importances}).sort_values("importance", ascending=False)

plt.figure(figsize=(6,4))
sns.barplot(data=fi, x="importance", y="feature", orient="h")
plt.title("Feature Importance")
plt.tight_layout()
plt.show()
```



Confusion Matrix
True label
Low
Medium
High
13
1
0
0
8
1
0
0
2
Predicted label
0
2
4
6
8
10
12



Feature Importance
feature
avg_paragraph_len
flesch_reading_ease
word_count
sentence_count
list_density
importance
0.00
0.05
0.10
0.15
0.20
0.25

Real-Time Analysis(analyze_url)

```
import requests, time, json
from sklearn.metrics.pairwise import cosine_similarity

# User-Agent header to avoid naive bot blocks
headers = {"User-Agent": "Mozilla/5.0 (compatible;
SEOContentDetector/1.0; +https://example.com/bot)"}

# Load model and reuse artifacts from notebook run
loaded_model = joblib.load(f"{base}/models/quality_model.pkl")
indexed_urls = lab_df['url'].tolist()
index_embeddings = embeddings

def scrape_url(url: str, timeout=12):
    """
    Fetch a URL with basic headers and return parsed title & body.
    """
    try:
        resp = requests.get(url, headers=headers, timeout=timeout)
        if resp.status_code != 200:
            return "", ""
        html = resp.text
        return extract_title(html), extract_main_text(html)
    except Exception:
        return "", ""

def compute_features(text: str):
    """
    Compute local features consistent with training pipeline.
    """
    ct = clean_text_basic(text)
    wc = len(ct.split())
    sc = sentence_count(text)
    fre = flesch_reading_ease_score(text)
    apl = approx_avg_paragraph_len(text)
    ld = list_density(text)
    return ct, wc, sc, fre, apl, ld

def vectorize_text(texts):
    """
    Reuse the sentence-transformer embedder loaded earlier.
    """
    return embedder.encode(texts, convert_to_numpy=True,
normalize_embeddings=True)

def find_similar(vec, top_k=5, threshold=0.75):
    """
    Retrieve most similar pages by cosine similarity against corpus
index.
```

```
    """
    sims = cosine_similarity(vec.reshape(1, -1),
index_embeddings).ravel()
    top_idx = np.argsort(sims)[-top_k:][::-1]
    return [{"url": indexed_urls[i], "similarity": float(sims[i])} for
i in top_idx if sims[i] >= threshold]

def rule_label(wc, fre):
    """
    Rule fallback consistent with labeling spec.
    """
    if (wc > 1500) and (50 <= fre <= 70): return "High"
    if (wc < 500) or (fre < 30): return "Low"
    return "Medium"

def analyze_url(url: str):
    """
    One-call analysis:
    - Scrape page with UA header and short delay
    - Extract features aligned with training
    - Embed and retrieve similar pages
    - Predict quality label (model or rule fallback)
    """
    time.sleep(1.2)  # polite delay
    title, body = scrape_url(url)
    ct, wc, sc, fre, apl, ld = compute_features(body)
    vec = vectorize_text([ct])[0]
    similar = find_similar(vec, top_k=5, threshold=0.75)
    X_one = pd.DataFrame([{
        "word_count": wc, "sentence_count": sc, "flesch_reading_ease":
fre,
        "avg_paragraph_len": apl, "list_density": ld
    }])
    try:
        pred = loaded_model.predict(X_one)[0]
    except Exception:
        pred = rule_label(wc, fre)
    return {
        "url": url, "title": title,
        "word_count": wc, "sentence_count": sc, "readability": fre,
        "avg_paragraph_len": apl, "list_density": ld,
        "quality_label": pred, "is_thin": bool(wc < 500),
        "similar_to": similar
    }

# Example use:
res = analyze_url("https://example.com/test-article")
print(json.dumps(res, indent=2))
```

```
{
  "url": "https://example.com/test-article",
  "title": "",
  "word_count": 0,
  "sentence_count": 0,
  "readability": 0.0,
  "avg_paragraph_len": 0.0,
  "list_density": 0.0,
  "quality_label": "Low",
  "is_thin": true,
  "similar_to": [
    {
      "url": "https://www.cnbc.com/artificial-intelligence/",
      "similarity": 1.0
    },
    {
      "url": "https://www.reuters.com/technology/artificial-
intelligence/",
      "similarity": 1.0
    },
    {
      "url": "https://www.bbc.com/news/topics/c404v061z99t",
      "similarity": 1.0
    },
    {
      "url":
"https://www.cloudflare.com/learning/access-management/what-is-ztna/",
      "similarity": 1.0
    },
    {
      "url": "https://www.connectwise.com/blog/phishing-prevention-
tips",
      "similarity": 1.0
    }
  ]
}
```

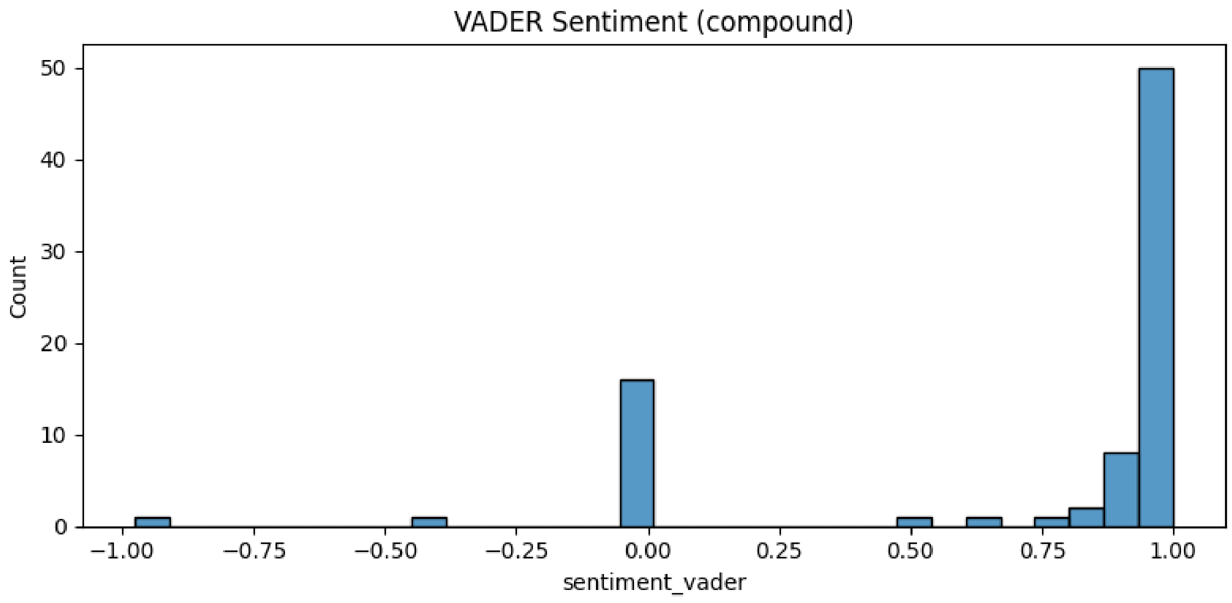
Advanced NLP: VADER sentiment, spaCy NER, BERTopic topics, and visuals

```python
from nltk.sentiment import SentimentIntensityAnalyzer
from collections import Counter
from bertopic import BERTopic

# Initialize VADER once
sia = SentimentIntensityAnalyzer()

def vader_compound(text: str) -> float:
    """
    Lexicon-based sentiment for web text; returns [-1, 1] compound
score.
    """
    try:
        return sia.polarity_scores(text or "")["compound"]
    except Exception:
        return 0.0

def ner_counts(text: str):
    """
    Count total entities and top 5 labels using spaCy
'en_core_web_sm';
    cap text length for latency control in notebooks.
    """
    try:
        doc = nlp((text or "")[:30000])
        labels = [ent.label_ for ent in doc.ents]
        total = len(labels)
        most = Counter(labels).most_common(5)
        return total, dict(most)
    except Exception:
        return 0, {}

feat_df['sentiment_vader'] =
feat_df['body_text'].apply(vader_compound)
ner_totals, ner_top_map = [], []
for t in feat_df['body_text']:
    tot, mp = ner_counts(t)
    ner_totals.append(tot)
    ner_top_map.append(mp)
feat_df['ner_total'] = ner_totals
feat_df['ner_top5'] = ner_top_map

# Topic modeling over sufficiently long docs; compute summaries
docs_for_topics = [txt for txt in feat_df['clean_text'].tolist() if
len(txt.split()) >= 50]
topic_model = BERTopic(language="english", verbose=False)
topics, probs = topic_model.fit_transform(docs_for_topics)
topic_freq = topic_model.get_topic_freq().sort_values("Count",
ascending=False).head(10)
topic_examples = {int(t): topic_model.get_topic(int(t)) for t in
topic_freq['Topic'].tolist() if int(t) != -1}

# Update features.csv with advanced columns for downstream & Streamlit
upload
feat_out_adv = feat_out.copy()
feat_out_adv['sentiment_vader'] = feat_df['sentiment_vader']
feat_out_adv['ner_total'] = feat_df['ner_total']
feat_out_adv['ner_top5'] = feat_df['ner_top5'].apply(lambda d: str(d))
```

```
feat_out_adv.to_csv(f"{base}/data/features.csv", index=False)
topic_freq.head(5)
```

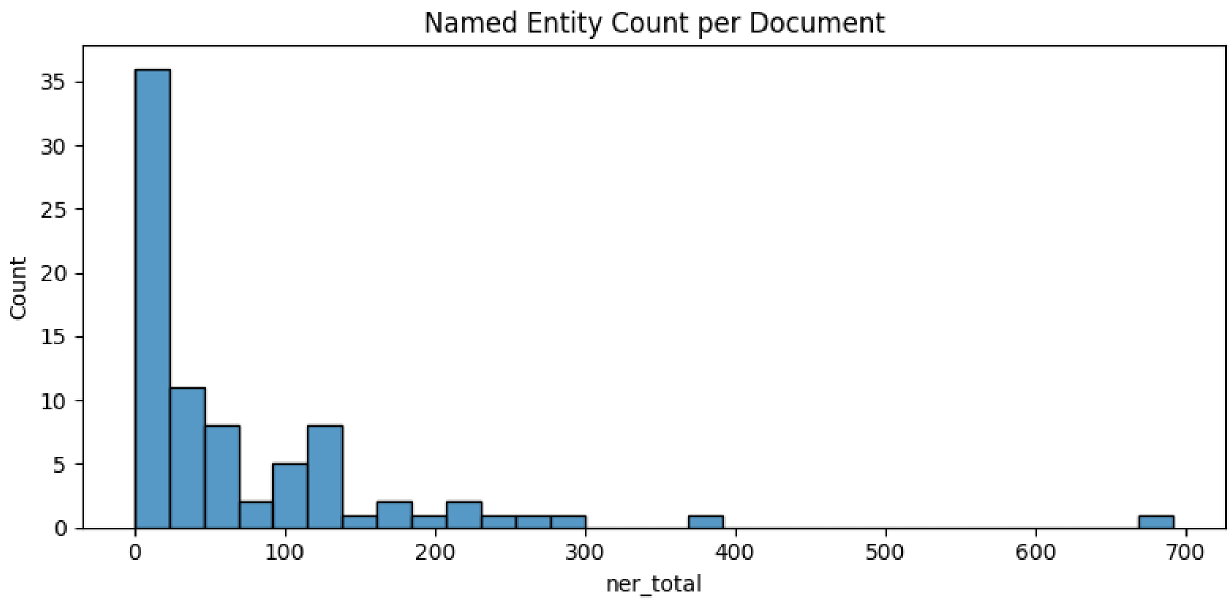
{"summary":"{\n \"name\": \"topic_freq\",\n \"rows\": 4,\n \"fields\": [\n   {\n   \"column\": \"Topic\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": -1,\n   \"max\": 2,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   1,\n   -1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 7,\n   \"min\": 8,\n   \"max\": 25,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   20,\n   8,\n   25\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"topic_freq"}

```
# Advanced NLP visuals
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud

plt.figure(figsize=(8,4))
sns.histplot(feat_df['sentiment_vader'], bins=30)
plt.title("VADER Sentiment (compound)")
plt.tight_layout()
plt.show()
```

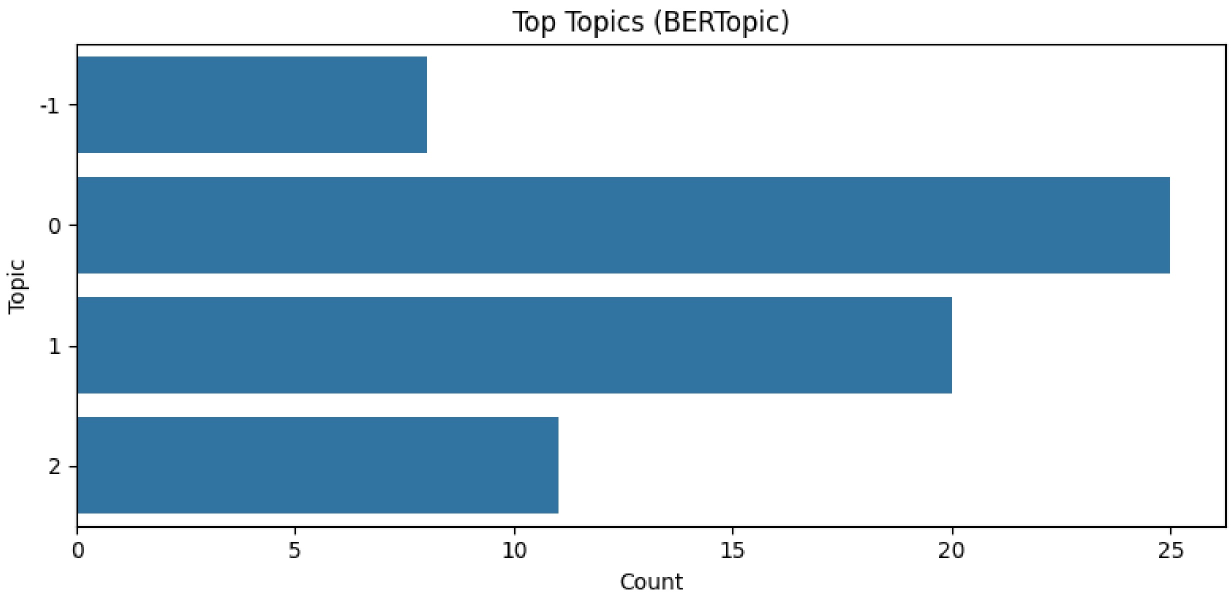


```
plt.figure(figsize=(8,4))
sns.histplot(feat_df['ner_total'], bins=30)
```

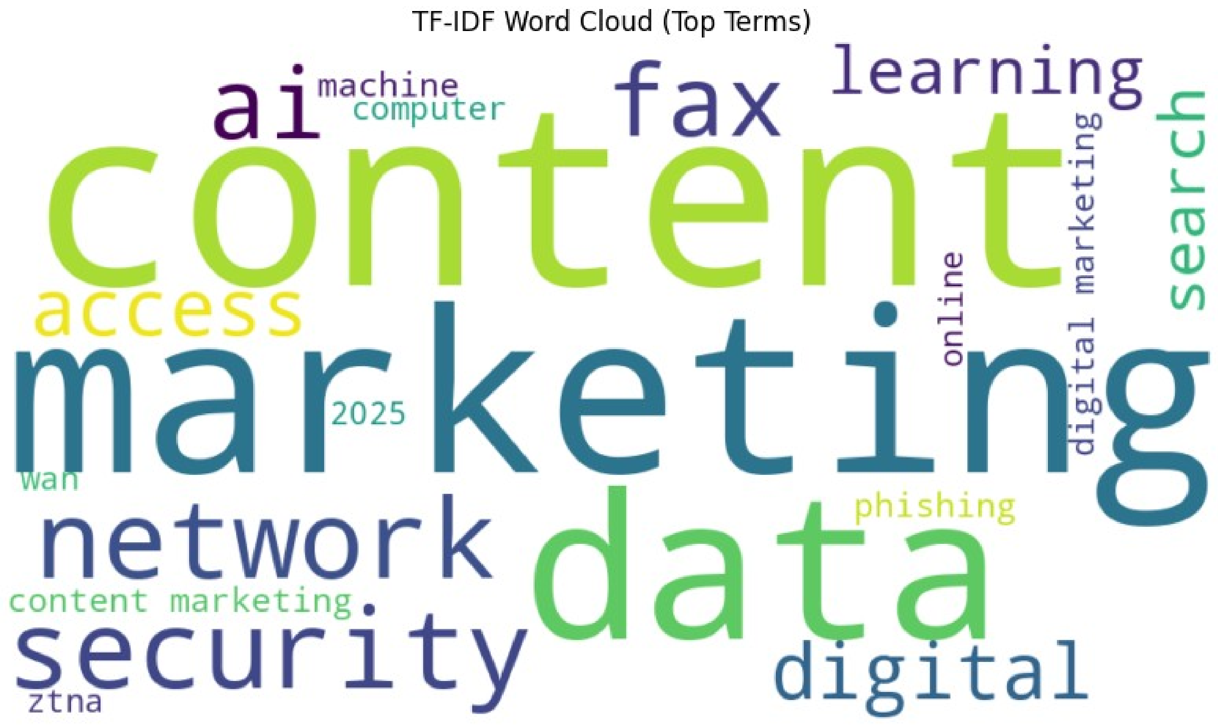
```
plt.title("Named Entity Count per Document")
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(8,4))
sns.barplot(data=topic_freq.head(10), x="Count", y="Topic",
orient="h")
plt.title("Top Topics (BERTopic)")
plt.tight_layout()
plt.show()
```



```
scores = dict(zip(term_df['term'], term_df['score']))
wc = WordCloud(width=900, height=500,
background_color="white").generate_from_frequencies(scores)
plt.figure(figsize=(10,6))
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.title("TF-IDF Word Cloud (Top Terms)")
plt.show()
```



Save requirements.txt, .gitignore, README template (with Streamlit URL placeholder)

```
req = """
streamlit==1.38.0
beautifulsoup4==4.12.3
lxml==5.3.0
textstat==0.7.4
scikit-learn==1.5.2
sentence-transformers==3.2.1
numpy==2.1.2
pandas==2.2.3
tqdm==4.66.5
nltk==3.9.1
unidecode==1.3.8
rapidfuzz==3.9.7
requests==2.32.3
```

```
joblib==1.4.2
matplotlib==3.9.2
seaborn==0.13.2
wordcloud==1.9.3
plotly==5.24.1
spacy==3.7.5
bertopic==0.16.2
umap-learn==0.5.6
hdbscan==0.8.37
datasketch==1.6.5
"""
with open(f"{base}/requirements.txt", "w") as f:
    f.write(req.strip())

gitignore = """
__pycache__/
*.pyc
*.pyo
*.pyd
.ipynb_checkpoints/
.venv/
venv/
env/
*.env
*.egg-info/
.DS_Store
*.model
*.pkl
"""
with open(f"{base}/.gitignore", "w") as f:
    f.write(gitignore.strip())

readme = """
# SEO Content Detector

End-to-end pipeline for SEO content quality scoring and duplicate
detection with a Colab notebook and optional Streamlit app.

## Setup
git clone https://github.com/yourusername/seo-content-detector
cd seo-content-detector
pip install -r requirements.txt
jupyter notebook notebooks/seo_pipeline.ipynb

## Quick Start
- Place data.csv under data/ with columns: url, html_content.
- Run the notebook cells.
- Outputs:
  - data/extracted_content.csv
  - data/features.csv
  - data/duplicates.csv
  - models/quality_model.pkl

## Key Decisions
- Parsing: <main>/<article> preferred; boilerplate stripped via tag
removal.
- Similarity: MinHash LSH for lexical candidates + cosine on MiniLM
embeddings; tuned thresholds.
- Classifier: RandomForest on content + structure features; baseline =
word_count rule.
- Advanced NLP: VADER sentiment, spaCy NER, BERTopic topics; TF-IDF
keyword improvements.

## Results Summary
- See classification report & confusion matrix in notebook.
- Duplicates: data/duplicates.csv and heatmap in notebook.
- Real-time: analyze_url(url) returns quality & similar pages.

## Limitations
- Extraction varies by site template and language.
- Readability is advisory; pair with structure and intent.
- LSH/cosine thresholds require corpus-specific tuning.

## Streamlit
- Deployed URL: https://YOUR-APP.streamlit.app
"""
with open(f"{base}/README.md", "w") as f:
    f.write(readme)
print("Wrote requirements.txt, .gitignore, README.md")
```

```
Wrote requirements.txt, .gitignore, README.md
```

Streamlit app with caching and advanced NLP, proper structure

```
import shutil

# Copy model for the app
shutil.copyfile(f"{base}/models/quality_model.pkl",
f"{base}/streamlit_app/models/quality_model.pkl")

# utils/parser.py
parser_py = r"""
import re, requests
from bs4 import BeautifulSoup
from unidecode import unidecode

headers = {"User-Agent": "Mozilla/5.0 (compatible;
SEOContentDetector/1.0; +https://example.com/bot)"}

def extract_main_text(html):
    try:
        soup = BeautifulSoup(html or "", "lxml")
        container = soup.find('main') or soup.find('article') or
soup.body or soup
        for tag in
container.find_all(['script','style','nav','footer','header','form','a
side']):
            tag.decompose()
        paragraphs = [p.get_text(" ", strip=True) for p in
container.find_all('p')]
        text = " ".join(paragraphs) if paragraphs else
container.get_text(" ", strip=True)
        text = re.sub(r'\s+', ' ', text).strip()
        return unidecode(text)
    except Exception:
        return ""

def extract_title(html):
    try:
        soup = BeautifulSoup(html or "", "lxml")
        if soup.title and soup.title.string:
            return soup.title.string.strip()
        og = soup.find("meta", property="og:title")
        if og and og.get("content"):
            return og.get("content").strip()
        h1 = soup.find("h1")
        if h1:
            return h1.get_text(" ", strip=True)
        return ""
    except Exception:
        return ""

def scrape_url(url: str, timeout=12):
    try:
        resp = requests.get(url, headers=headers, timeout=timeout)
        if resp.status_code != 200:
            return "", ""
        return extract_title(resp.text), extract_main_text(resp.text)
    except Exception:
        return "", ""
"""
with open(f"{base}/streamlit_app/utils/parser.py", "w") as f:
    f.write(parser_py)

# utils/features.py with caching-ready design (Streamlit caches in
app.py)
features_py = r"""
import re, numpy as np, textstat, spacy
from nltk.tokenize import sent_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
from sentence_transformers import SentenceTransformer
from collections import Counter

# Load spaCy & VADER once at module import; in Streamlit, use cache in
app for heavy models if needed
try:
    nlp = spacy.load("en_core_web_sm")
except:
    import spacy.cli as spcli
    spcli.download("en_core_web_sm")
    nlp = spacy.load("en_core_web_sm")

sia = SentimentIntensityAnalyzer()

def clean_text_basic(s: str) -> str:
    s = (s or "").lower().strip()
    s = re.sub(r'\s+', ' ', s)
    return s

def sentence_count(s: str) -> int:
    try:
        return len(sent_tokenize(s)) if s else 0
    except Exception:
        return 0

def flesch_reading_ease(s: str) -> float:
    try:
        return float(textstat.flesch_reading_ease(s)) if s and
len(s.split()) >= 5 else 0.0
    except Exception:
        return 0.0

def vader_compound(text: str) -> float:
    try:
        return sia.polarity_scores(text or '')['compound']
    except Exception:
        return 0.0

def ner_top(text: str):
    try:
        doc = nlp((text or "")[:30000])
        labels = [ent.label_ for ent in doc.ents]
        return len(labels), dict(Counter(labels).most_common(5))
    except Exception:
        return 0, {}

def load_embedder():
    return SentenceTransformer("all-MiniLM-L6-v2")

def vectorize_texts(embedder, texts):
    return embedder.encode(texts, convert_to_numpy=True,
normalize_embeddings=True)
"""
with open(f"{base}/streamlit_app/utils/features.py", "w") as f:
    f.write(features_py)

# utils/scorer.py
scorer_py = r"""
import numpy as np
import joblib
from sklearn.metrics.pairwise import cosine_similarity

def load_quality_model(path: str):
    return joblib.load(path)

def find_similar_factory(index_embeddings, index_urls):
    def _find(vec, top_k=5, threshold=0.75):
        if index_embeddings is None or len(index_urls) == 0:
            return []
        sims = cosine_similarity(vec.reshape(1, -1),
index_embeddings).ravel()
        top_idx = sims.argsort()[-top_k:][::-1]
        return [{"url": index_urls[i], "similarity": float(sims[i])}
for i in top_idx if sims[i] >= threshold]
    return _find
"""
with open(f"{base}/streamlit_app/utils/scorer.py", "w") as f:
    f.write(scorer_py)

# app.py with caching (st.cache_resource / st.cache_data)
app_py = r"""
import streamlit as st
import pandas as pd
import numpy as np
import json
from utils.parser import scrape_url
from utils.features import (clean_text_basic, sentence_count,
flesch_reading_ease,
                            load_embedder, vectorize_texts,
vader_compound, ner_top)
from utils.scorer import load_quality_model, find_similar_factory

st.set_page_config(page_title="SEO Content Detector", layout="wide")
st.title("SEO Content Detector")
st.write("Analyze quality, duplicates, readability, sentiment, and
entities.")

# Cache heavy resources for faster subsequent runs
@st.cache_resource
def get_model():
    return load_quality_model("models/quality_model.pkl")

@st.cache_resource
def get_embedder():
    return load_embedder()

model = get_model()
embedder = get_embedder()

uploaded_feats = st.file_uploader("Optional: Upload features.csv (to
enable similarity to known pages)", type=["csv"])

index_urls, index_embeddings = [], None
if uploaded_feats:
    feats = pd.read_csv(uploaded_feats)
    emb = feats['embedding'].apply(lambda s: np.array(json.loads(s)))
    index_embeddings = np.vstack(emb.values)
    index_urls = feats['url'].tolist()

find_similar = find_similar_factory(index_embeddings, index_urls)

url = st.text_input("Enter a URL to analyze")
if st.button("Analyze") and url:
    title, body = scrape_url(url)
    ct = clean_text_basic(body)
    wc = len(ct.split()); sc = sentence_count(body); fre =
flesch_reading_ease(body)
    sent = vader_compound(body)
    ner_total, ner_top5 = ner_top(body)
    vec = vectorize_texts(embedder, [ct])[0]
    similar = find_similar(vec, top_k=5, threshold=0.75) if
index_embeddings is not None else []

    X_one = pd.DataFrame([{"word_count": wc, "sentence_count": sc,
"flesch_reading_ease": fre,
                           "avg_paragraph_len":
float(len(ct.split())/(sc or 1)), "list_density": 0.0}])
    try:
        pred = model.predict(X_one)[0]
    except:
        pred = "Medium"

    col1, col2 = st.columns(2)
    with col1:
        st.subheader("Core Metrics")
        st.json({
            "url": url, "title": title, "word_count": wc,
"sentence_count": sc,
            "readability": fre, "quality_label": pred, "is_thin":
bool(wc < 500)
```

```
        })
    with col2:
        st.subheader("Advanced NLP")
        st.json({
            "sentiment_vader_compound": sent,
            "ner_total": ner_total,
            "ner_top5": ner_top5
        })

    st.subheader("Similar Pages")
    st.json(similar if similar else [{"info":"Upload features.csv to
enable similarity search"}])
"""
with open(f"{base}/streamlit_app/app.py", "w") as f:
    f.write(app_py)

print("Streamlit app created under streamlit_app/ with caching.")
```

```
Streamlit app created under streamlit_app/ with caching.
```

Final Checklist

```
print("Data:", os.listdir(f"{base}/data"))
print("Models:", os.listdir(f"{base}/models"))
print("Streamlit app:", os.listdir(f"{base}/streamlit_app"))
print("Streamlit utils:", os.listdir(f"{base}/streamlit_app/utils"))

# NEXT STEPS:
# - Commit seo-content-detector/ to GitHub.
# - In Streamlit Cloud, set Main file path: streamlit_app/app.py and
deploy.
# - Update README.md with the deployed URL.
```

```
Data: ['extracted_content.csv', 'data.csv', 'features.csv',
'duplicates.csv']
Models: ['quality_model.pkl']
Streamlit app: ['utils', 'models', 'app.py']
Streamlit utils: ['features.py', 'parser.py', 'scorer.py']
```

Zip Structural Download

```
# Zip the entire project folder and download it locally
import shutil, os
base = "seo-content-detector"   # folder created
zip_name = f"{base}.zip"
# Remove old zip if it exists
if os.path.exists(zip_name):
    os.remove(zip_name)
# Create zip
shutil.make_archive(base, 'zip', base_dir=base)
```

```
# Download
from google.colab import files
files.download(zip_name)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```