Lab5

In this lab, we will explore how to parse documents in various formats and store the extracted text in a vector database.

Before starting the lab, you need to download "The_Daily_Drucker_January.pdf" and place it in the same directory as this Notebook before proceeding.

1. Environment setup

Before running the following code, ensure you have completed these tutorials:

1. Llama Cloud API for LlamaParse
2. FireCrawl API
3. PostgreSQL Connection Setup

In [1]:
```
# Install required packages
# use -U (ugrade mode) to ensure the package is upgraded to the lastest v
%pip install -U langchain_community pymupdf llama-cloud-services llama-in
%pip install -U firecrawl-py langchain_community
```

```
Requirement already satisfied: langchain_community in ./lib/python3.10/sit
e-packages (0.3.19)
Requirement already satisfied: pymupdf in ./lib/python3.10/site-packages
(1.25.3)
Requirement already satisfied: llama-cloud-services in ./lib/python3.10/si
te-packages (0.6.5)
Requirement already satisfied: llama-index in ./lib/python3.10/site-packag
es (0.12.23)
Requirement already satisfied: llama-index-core in ./lib/python3.10/site-p
ackages (0.12.24)
Requirement already satisfied: langchain_postgres in ./lib/python3.10/site
-packages (0.0.13)
Requirement already satisfied: requests<3,>=2 in ./lib/python3.10/site-pac
kages (from langchain_community) (2.32.3)
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in ./lib/python
3.10/site-packages (from langchain_community) (8.5.0)
Requirement already satisfied: PyYAML>=5.3 in ./lib/python3.10/site-packag
es (from langchain_community) (6.0.2)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in ./lib/pytho
n3.10/site-packages (from langchain_community) (0.6.7)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in ./lib/python3.10/s
ite-packages (from langchain_community) (3.11.11)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in ./lib/py
thon3.10/site-packages (from langchain_community) (2.7.1)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.41 in ./lib/pyth
on3.10/site-packages (from langchain_community) (0.3.44)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in ./lib/python3.1
0/site-packages (from langchain_community) (0.4.0)
Requirement already satisfied: langchain<1.0.0,>=0.3.20 in ./lib/python3.1
0/site-packages (from langchain_community) (0.3.20)
Requirement already satisfied: langsmith<0.4,>=0.1.125 in ./lib/python3.1
0/site-packages (from langchain_community) (0.3.13)
Requirement already satisfied: numpy<3,>=1.26.2 in ./lib/python3.10/site-p
ackages (from langchain_community) (1.26.4)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in ./lib/python3.10/site
-packages (from langchain_community) (2.0.37)
Requirement already satisfied: llama-cloud<0.2.0,>=0.1.14 in ./lib/python
3.10/site-packages (from llama-cloud-services) (0.1.14)
Requirement already satisfied: pydantic!=2.10 in ./lib/python3.10/site-pac
kages (from llama-cloud-services) (2.10.6)
Requirement already satisfied: click<9.0.0,>=8.1.7 in ./lib/python3.10/sit
e-packages (from llama-cloud-services) (8.1.8)
Requirement already satisfied: python-dotenv<2.0.0,>=1.0.1 in ./lib/python
3.10/site-packages (from llama-cloud-services) (1.0.1)
Requirement already satisfied: nltk>3.8.1 in ./lib/python3.10/site-package
s (from llama-index) (3.9.1)
Requirement already satisfied: llama-index-agent-openai<0.5.0,>=0.4.0 in
./lib/python3.10/site-packages (from llama-index) (0.4.6)
Requirement already satisfied: llama-index-program-openai<0.4.0,>=0.3.0 in
./lib/python3.10/site-packages (from llama-index) (0.3.1)
Requirement already satisfied: llama-index-readers-llama-parse>=0.4.0 in
./lib/python3.10/site-packages (from llama-index) (0.4.0)
Requirement already satisfied: llama-index-cli<0.5.0,>=0.4.1 in ./lib/pyth
on3.10/site-packages (from llama-index) (0.4.1)
Requirement already satisfied: llama-index-embeddings-openai<0.4.0,>=0.3.0
in ./lib/python3.10/site-packages (from llama-index) (0.3.1)
Requirement already satisfied: llama-index-multi-modal-llms-openai<0.5.0,>
=0.4.0 in ./lib/python3.10/site-packages (from llama-index) (0.4.3)
Requirement already satisfied: llama-index-readers-file<0.5.0,>=0.4.0 in
./lib/python3.10/site-packages (from llama-index) (0.4.6)
```

```
Requirement already satisfied: llama-index-indices-managed-llama-cloud>=0.
4.0 in ./lib/python3.10/site-packages (from llama-index) (0.6.8)
Requirement already satisfied: llama-index-llms-openai<0.4.0,>=0.3.0 in ./
lib/python3.10/site-packages (from llama-index) (0.3.25)
Requirement already satisfied: llama-index-question-gen-openai<0.4.0,>=0.
3.0 in ./lib/python3.10/site-packages (from llama-index) (0.3.0)
Requirement already satisfied: networkx>=3.0 in ./lib/python3.10/site-pack
ages (from llama-index-core) (3.4.2)
Requirement already satisfied: wrapt in ./lib/python3.10/site-packages (fr
om llama-index-core) (1.17.2)
Requirement already satisfied: tiktoken>=0.3.3 in ./lib/python3.10/site-pa
ckages (from llama-index-core) (0.8.0)
Requirement already satisfied: filetype<2.0.0,>=1.2.0 in ./lib/python3.10/
site-packages (from llama-index-core) (1.2.0)
Requirement already satisfied: typing-inspect>=0.8.0 in ./lib/python3.10/s
ite-packages (from llama-index-core) (0.9.0)
Requirement already satisfied: nest-asyncio<2.0.0,>=1.5.8 in ./lib/python
3.10/site-packages (from llama-index-core) (1.6.0)
Requirement already satisfied: typing-extensions>=4.5.0 in ./lib/python3.1
0/site-packages (from llama-index-core) (4.12.2)
Requirement already satisfied: tqdm<5.0.0,>=4.66.1 in ./lib/python3.10/sit
e-packages (from llama-index-core) (4.67.1)
Requirement already satisfied: deprecated>=1.2.9.3 in ./lib/python3.10/sit
e-packages (from llama-index-core) (1.2.18)
Requirement already satisfied: dirtyjson<2.0.0,>=1.0.8 in ./lib/python3.1
0/site-packages (from llama-index-core) (1.0.8)
Requirement already satisfied: fsspec>=2023.5.0 in ./lib/python3.10/site-p
ackages (from llama-index-core) (2025.3.0)
Requirement already satisfied: pillow>=9.0.0 in ./lib/python3.10/site-pack
ages (from llama-index-core) (11.1.0)
Requirement already satisfied: httpx in ./lib/python3.10/site-packages (fr
om llama-index-core) (0.28.1)
Requirement already satisfied: psycopg-pool<4.0.0,>=3.2.1 in ./lib/python
3.10/site-packages (from langchain_postgres) (3.2.6)
Requirement already satisfied: pgvector<0.4 in ./lib/python3.10/site-packa
ges (from langchain_postgres) (0.3.6)
Requirement already satisfied: psycopg<4,>=3 in ./lib/python3.10/site-pack
ages (from langchain_postgres) (3.2.5)
Requirement already satisfied: attrs>=17.3.0 in ./lib/python3.10/site-pack
ages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (25.1.0)
Requirement already satisfied: async-timeout<6.0,>=4.0 in ./lib/python3.1
0/site-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (4.0.3)
Requirement already satisfied: propcache>=0.2.0 in ./lib/python3.10/site-p
ackages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (0.2.1)
Requirement already satisfied: aiosignal>=1.1.2 in ./lib/python3.10/site-p
ackages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.3.2)
Requirement already satisfied: frozenlist>=1.1.1 in ./lib/python3.10/site-
packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.5.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in ./lib/python3.10/site-
packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.18.3)
Requirement already satisfied: multidict<7.0,>=4.5 in ./lib/python3.10/sit
e-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (6.1.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in ./lib/python3.1
0/site-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (2.4.4)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in ./lib/python
3.10/site-packages (from dataclasses-json<0.7,>=0.5.7->langchain_communit
y) (3.26.1)
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in
./lib/python3.10/site-packages (from langchain<1.0.0,>=0.3.20->langchain_c
ommunity) (0.3.6)
```

```
Requirement already satisfied: jsonpatch<2.0,>=1.33 in ./lib/python3.10/si
te-packages (from langchain-core<1.0.0,>=0.3.41->langchain_community) (1.3
3)
Requirement already satisfied: packaging<25,>=23.2 in ./lib/python3.10/sit
e-packages (from langchain-core<1.0.0,>=0.3.41->langchain_community) (23.
2)
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in ./lib/python3.
10/site-packages (from langsmith<0.4,>=0.1.125->langchain_community) (0.2
3.0)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in ./lib/python3.10/s
ite-packages (from langsmith<0.4,>=0.1.125->langchain_community) (3.10.15)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in ./lib/py
thon3.10/site-packages (from langsmith<0.4,>=0.1.125->langchain_community)
(1.0.0)
Requirement already satisfied: anyio in ./lib/python3.10/site-packages (fr
om httpx->llama-index-core) (4.8.0)
Requirement already satisfied: idna in ./lib/python3.10/site-packages (fro
m httpx->llama-index-core) (3.10)
Requirement already satisfied: certifi in ./lib/python3.10/site-packages
(from httpx->llama-index-core) (2025.1.31)
Requirement already satisfied: httpcore==1.* in ./lib/python3.10/site-pack
ages (from httpx->llama-index-core) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in ./lib/python3.10/site-pa
ckages (from httpcore==1.*->httpx->llama-index-core) (0.14.0)
Requirement already satisfied: openai>=1.14.0 in ./lib/python3.10/site-pac
kages (from llama-index-agent-openai<0.5.0,>=0.4.0->llama-index) (1.61.1)
Requirement already satisfied: pandas in ./lib/python3.10/site-packages (f
rom llama-index-readers-file<0.5.0,>=0.4.0->llama-index) (2.2.3)
Requirement already satisfied: striprtf<0.0.27,>=0.0.26 in ./lib/python3.1
0/site-packages (from llama-index-readers-file<0.5.0,>=0.4.0->llama-index)
(0.0.26)
Requirement already satisfied: beautifulsoup4<5.0.0,>=4.12.3 in ./lib/pyth
on3.10/site-packages (from llama-index-readers-file<0.5.0,>=0.4.0->llama-i
ndex) (4.13.3)
Requirement already satisfied: pypdf<6.0.0,>=5.1.0 in ./lib/python3.10/sit
e-packages (from llama-index-readers-file<0.5.0,>=0.4.0->llama-index) (5.
3.1)
Requirement already satisfied: llama-parse>=0.5.0 in ./lib/python3.10/site
-packages (from llama-index-readers-llama-parse>=0.4.0->llama-index) (0.6.
4.post1)
Requirement already satisfied: regex>=2021.8.3 in ./lib/python3.10/site-pa
ckages (from nltk>3.8.1->llama-index) (2024.11.6)
Requirement already satisfied: joblib in ./lib/python3.10/site-packages (f
rom nltk>3.8.1->llama-index) (1.4.2)
Requirement already satisfied: pydantic-core==2.27.2 in ./lib/python3.10/s
ite-packages (from pydantic!=2.10->llama-cloud-services) (2.27.2)
Requirement already satisfied: annotated-types>=0.6.0 in ./lib/python3.10/
site-packages (from pydantic!=2.10->llama-cloud-services) (0.7.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./lib/python3.10/site
-packages (from requests<3,>=2->langchain_community) (2.3.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./lib/python3.1
0/site-packages (from requests<3,>=2->langchain_community) (3.4.1)
Requirement already satisfied: greenlet!=0.4.17 in ./lib/python3.10/site-p
ackages (from SQLAlchemy<3,>=1.4->langchain_community) (3.1.1)
Requirement already satisfied: mypy-extensions>=0.3.0 in ./lib/python3.10/
site-packages (from typing-inspect>=0.8.0->llama-index-core) (1.0.0)
Requirement already satisfied: soupsieve>1.2 in ./lib/python3.10/site-pack
ages (from beautifulsoup4<5.0.0,>=4.12.3->llama-index-readers-file<0.5.0,>
=0.4.0->llama-index) (2.6)
Requirement already satisfied: jsonpointer>=1.9 in ./lib/python3.10/site-p
```

```
ackages (from jsonpatch<2.0,>=1.33->langchain-core<1.0.0,>=0.3.41->langcha
in_community) (3.0.0)
Requirement already satisfied: jiter<1,>=0.4.0 in ./lib/python3.10/site-pa
ckages (from openai>=1.14.0->llama-index-agent-openai<0.5.0,>=0.4.0->llama
-index) (0.8.2)
Requirement already satisfied: distro<2,>=1.7.0 in ./lib/python3.10/site-p
ackages (from openai>=1.14.0->llama-index-agent-openai<0.5.0,>=0.4.0->llam
a-index) (1.9.0)
Requirement already satisfied: sniffio in ./lib/python3.10/site-packages
(from openai>=1.14.0->llama-index-agent-openai<0.5.0,>=0.4.0->llama-index)
(1.3.1)
Requirement already satisfied: exceptiongroup>=1.0.2 in ./lib/python3.10/s
ite-packages (from anyio->httpx->llama-index-core) (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in ./lib/python3.10/
site-packages (from pandas->llama-index-readers-file<0.5.0,>=0.4.0->llama-
index) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./lib/python3.10/site-packa
ges (from pandas->llama-index-readers-file<0.5.0,>=0.4.0->llama-index) (20
25.1)
Requirement already satisfied: tzdata>=2022.7 in ./lib/python3.10/site-pac
kages (from pandas->llama-index-readers-file<0.5.0,>=0.4.0->llama-index)
(2025.1)
Requirement already satisfied: six>=1.5 in ./lib/python3.10/site-packages
(from python-dateutil>=2.8.2->pandas->llama-index-readers-file<0.5.0,>=0.
4.0->llama-index) (1.17.0)

[notice] A new release of pip is available: 23.0.1 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: firecrawl-py in ./lib/python3.10/site-packa
ges (1.13.5)
Requirement already satisfied: langchain_community in ./lib/python3.10/sit
e-packages (0.3.19)
Requirement already satisfied: pydantic>=2.10.3 in ./lib/python3.10/site-p
ackages (from firecrawl-py) (2.10.6)
Requirement already satisfied: python-dotenv in ./lib/python3.10/site-pack
ages (from firecrawl-py) (1.0.1)
Requirement already satisfied: websockets in ./lib/python3.10/site-package
s (from firecrawl-py) (15.0.1)
Requirement already satisfied: requests in ./lib/python3.10/site-packages
(from firecrawl-py) (2.32.3)
Requirement already satisfied: nest-asyncio in ./lib/python3.10/site-packa
ges (from firecrawl-py) (1.6.0)
Requirement already satisfied: numpy<3,>=1.26.2 in ./lib/python3.10/site-p
ackages (from langchain_community) (1.26.4)
Requirement already satisfied: PyYAML>=5.3 in ./lib/python3.10/site-packag
es (from langchain_community) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in ./lib/python3.10/site
-packages (from langchain_community) (2.0.37)
Requirement already satisfied: langchain<1.0.0,>=0.3.20 in ./lib/python3.1
0/site-packages (from langchain_community) (0.3.20)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in ./lib/python3.10/s
ite-packages (from langchain_community) (3.11.11)
Requirement already satisfied: langsmith<0.4,>=0.1.125 in ./lib/python3.1
0/site-packages (from langchain_community) (0.3.13)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in ./lib/pytho
n3.10/site-packages (from langchain_community) (0.6.7)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in ./lib/python3.1
0/site-packages (from langchain_community) (0.4.0)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.41 in ./lib/pyth
```

```
on3.10/site-packages (from langchain_community) (0.3.44)
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in ./lib/python
3.10/site-packages (from langchain_community) (8.5.0)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in ./lib/py
thon3.10/site-packages (from langchain_community) (2.7.1)
Requirement already satisfied: attrs>=17.3.0 in ./lib/python3.10/site-pack
ages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in ./lib/python3.10/site-
packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.5.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in ./lib/python3.10/site-
packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.18.3)
Requirement already satisfied: aiosignal>=1.1.2 in ./lib/python3.10/site-p
ackages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (1.3.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in ./lib/python3.1
0/site-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (2.4.4)
Requirement already satisfied: propcache>=0.2.0 in ./lib/python3.10/site-p
ackages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (0.2.1)
Requirement already satisfied: async-timeout<6.0,>=4.0 in ./lib/python3.1
0/site-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (4.0.3)
Requirement already satisfied: multidict<7.0,>=4.5 in ./lib/python3.10/sit
e-packages (from aiohttp<4.0.0,>=3.8.3->langchain_community) (6.1.0)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in ./lib/python
3.10/site-packages (from dataclasses-json<0.7,>=0.5.7->langchain_communit
y) (3.26.1)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in ./lib/python3.1
0/site-packages (from dataclasses-json<0.7,>=0.5.7->langchain_community)
(0.9.0)
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in
./lib/python3.10/site-packages (from langchain<1.0.0,>=0.3.20->langchain_c
ommunity) (0.3.6)
Requirement already satisfied: typing-extensions>=4.7 in ./lib/python3.10/
site-packages (from langchain-core<1.0.0,>=0.3.41->langchain_community)
(4.12.2)
Requirement already satisfied: packaging<25,>=23.2 in ./lib/python3.10/sit
e-packages (from langchain-core<1.0.0,>=0.3.41->langchain_community) (23.
2)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in ./lib/python3.10/si
te-packages (from langchain-core<1.0.0,>=0.3.41->langchain_community) (1.3
3)
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in ./lib/python3.
10/site-packages (from langsmith<0.4,>=0.1.125->langchain_community) (0.2
3.0)
Requirement already satisfied: httpx<1,>=0.23.0 in ./lib/python3.10/site-p
ackages (from langsmith<0.4,>=0.1.125->langchain_community) (0.28.1)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in ./lib/py
thon3.10/site-packages (from langsmith<0.4,>=0.1.125->langchain_community)
(1.0.0)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in ./lib/python3.10/s
ite-packages (from langsmith<0.4,>=0.1.125->langchain_community) (3.10.15)
Requirement already satisfied: annotated-types>=0.6.0 in ./lib/python3.10/
site-packages (from pydantic>=2.10.3->firecrawl-py) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.2 in ./lib/python3.10/s
ite-packages (from pydantic>=2.10.3->firecrawl-py) (2.27.2)
Requirement already satisfied: charset-normalizer<4,>=2 in ./lib/python3.1
0/site-packages (from requests->firecrawl-py) (3.4.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./lib/python3.10/site
-packages (from requests->firecrawl-py) (2.3.0)
Requirement already satisfied: idna<4,>=2.5 in ./lib/python3.10/site-packa
ges (from requests->firecrawl-py) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in ./lib/python3.10/site
```

```
-packages (from requests->firecrawl-py) (2025.1.31)
Requirement already satisfied: anyio in ./lib/python3.10/site-packages (fr
om httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain_community) (4.8.0)
Requirement already satisfied: httpcore==1.* in ./lib/python3.10/site-pack
ages (from httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain_community)
(1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in ./lib/python3.10/site-pa
ckages (from httpcore==1.*->httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->lan
gchain_community) (0.14.0)
Requirement already satisfied: jsonpointer>=1.9 in ./lib/python3.10/site-p
ackages (from jsonpatch<2.0,>=1.33->langchain-core<1.0.0,>=0.3.41->langcha
in_community) (3.0.0)
Requirement already satisfied: mypy-extensions>=0.3.0 in ./lib/python3.10/
site-packages (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7
->langchain_community) (1.0.0)
Requirement already satisfied: exceptiongroup>=1.0.2 in ./lib/python3.10/s
ite-packages (from anyio->httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langc
hain_community) (1.2.2)
Requirement already satisfied: sniffio>=1.1 in ./lib/python3.10/site-packa
ges (from anyio->httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain_comm
unity) (1.3.1)

[notice] A new release of pip is available: 23.0.1 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```python
from langchain_community.document_loaders import PyMuPDFLoader
from IPython.display import Markdown
from pprint import pprint  # Import pprint for better metadata readabilit
import textwrap
import os
from dotenv import load_dotenv

from langchain_openai import OpenAIEmbeddings
from langchain_community.document_loaders.firecrawl import FireCrawlLoade
```

In [3]:
```python
# Load environment variables from .env file
load_dotenv()

# Access the environment variable
openai_api_key = os.getenv("OPENAI_API_KEY")
connection_string = os.getenv("DB_CONNECTION")
llama_cloud_api_key =os.getenv("LLAMA_CLOUD_API_KEY")
firecrawl_api_key = os.getenv("FIRECRAWL_API_KEY")

# Quick check environment variables
if not os.getenv("OPENAI_API_KEY") or not os.getenv("DB_CONNECTION") or n
    print(f"Error: Missing one or more required environment variables") #
else:
    print("All environment variables loaded successfully")
```

```
All environment variables loaded successfully
```

2. Data Ingestion

The main goal of data ingestion is to gather, extract, and normalize raw text from multiple sources into a structured format before further preprocessing, such as

cleaning, chunking, embedding, and storing in a database. It includes the following steps

- Extract data from PDFs, DOCX, Web pages, csv, or databases.
- Covert data into plain texts.
- Add metadata such as file name, date, author, category.

LangChain allows users to parse various sources, including **PDF**, **CSV**, **Text**, **Webpages**, and more.

2.1. Parsing PDFs with LangChain

Let's explore PDF parsing using `PyMuPDFLoader`, a lightweight and efficient text extraction tool in LangChain. It is built upon `PyMuPDF` (also known as `Fitz`), a high-performance Python library for text extraction, search, and document manipulation across multiple formats, including PDF, XPS, OpenXPS, CBZ, and EPUB.

In this lab, we will learn how to use `PyMuPDFLoader` to efficiently extract text from PDFs. As a demonstration, we will work with an excerpt from one of Peter Drucker's well-known books, specifically the Daily Drucker January chapter.

Other Recommended PDF Parsing Tools For more advanced PDF parsing capabilities, consider the following tools:

1. `LlamaParse` – A robust document parsing tool by LlamaIndex, designed for high-quality extraction for complex PDFs. Learn more at LlamaParse.
2. `MinerU` – An open-source tool for high-quality data extraction, capable of converting PDFs to **Markdown** and **JSON** formats. Learn more at MinerU.

In [4]:
```
# Specify the path to the PDF file that located at the same location
pdf_file_path = "data_labs/The_Daily_Drucker_January.pdf"

# Get the file name without extension
pdf_name = os.path.splitext(os.path.basename(pdf_file_path))[0]
print(pdf_name)

# Create a PyMuPDFLoader instance with the specified file path
loader = PyMuPDFLoader(file_path = pdf_file_path,
                       mode='page')

# Load data into Document objects
docs = loader.load()
```

```
The_Daily_Drucker_January
```

In [4]:
```
# LangChain Document object type
type(docs)
```

Out[4]:
```
list
```

In [5]:
```
# Let's explore the Document object structure
print("Document collection size:", len(docs))
```

```
# Displaying the structure of the first document
doc = docs[0]

print("\nLangChain Document Object Structure:")
print(f"Available attributes: {list(vars(doc).keys())}")

# Focus on the two most important attributes
print("\nKey Attributes:")

# 1. Metadata - Contains document information like source file, page numb
print("1. metadata:")
pprint(f"   {doc.metadata}")

# 2. Page Content - The actual text content
print("\n2. page_content (excerpt):")
content_preview = doc.page_content[:300] + "..." if len(doc.page_content)
print(f"   {content_preview}")
```

```
Document collection size: 61

LangChain Document Object Structure:
Available attributes: ['id', 'metadata', 'page_content', 'type']

Key Attributes:
1. metadata:
("   {'producer': 'HarperCollins Publishers', 'creator': 'Harper Busines
s', "
 "'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "'The_Daily_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
"
 "'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
 "Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
 "'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
 "'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
 "'creationDate': 'D:20040101000000', 'page': 0}")

2. page_content (excerpt):
   1 January
Integrity in Leadership
The spirit of an organization is created from the top.
T he proof of the sincerity and seriousness of a management is
uncompromising emphasis on integrity of character. This, above all, has to
be
symbolized in management's "people" decisions. For it is character thr...
```

In [6]:
```
import random

#  Number of pages (max 3) to check
num_pages = min(3, len(docs))  # Ensure not exceeding available pages

# Randomly select 'num_pages'
sampled_pages = random.sample(range(len(docs)), num_pages)  # Selects 'nu

# Iterate through the randomly selected pages and print their content
for i in sampled_pages:
    print(f"\n{'='*40} PAGE {i+1} {'='*40}\n")  # Display page number
    print(f"Page Content: {docs[i].page_content}")  # Print page content
    pprint(f"Metadata: {docs[i].metadata}")  # Print metadata associated
    print("\n" + "="*90 + "\n")  # Separator for readability
```

========== PAGE 24 ==========

Page Content:
("Metadata: {'producer': 'HarperCollins Publishers', 'creator': 'Harper "
 "Business', 'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "'The_Daily_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
"
 "'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
 "Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
 "'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
 "'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
 "'creationDate': 'D:20040101000000', 'page': 23}")

==========

========== PAGE 59 ==========

Page Content: 30 January
Terrorism and Basic Trends
Management of an institution has to be grounded in basic and predictable t
rends that
persist regardless of today’s headlines.
T he terrorist attacks of September 2001 and America’s response to them
have profoundly changed world politics. We clearly face years of disorder,
especially in the Middle East. Management of an institution—whether a
business, a university, a hospital—has to be grounded in basic and
predictable trends that persist regardless of today’s headlines. It has to
exploit these trends as opportunities. And these basic trends are the
emergence of the Next Society and its new and unprecedented
characteristics, especially
the global shrinking of the youth population and the emergence of the “new
workforce”
the steady decline of manufacturing as a producer of wealth and jobs
the changes in the form, the structure, and the function of the corporatio
n and of its
top management
In times of great and unpredictable surprises, even basing one’s strategy
and
one’s policies on these unchanging and basic trends does not automatically
ensure success. But not to do so guarantees failure.

ACTION POINT: Write down three basic social trends that your business is
based on. Are these trends still intact?
Managing in the Next Society
("Metadata: {'producer': 'HarperCollins Publishers', 'creator': 'Harper "
 "Business', 'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "'The_Daily_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
"
 "'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
 "Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
 "'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
 "'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
 "'creationDate': 'D:20040101000000', 'page': 58}")

```
================================================================================
===============


======================================== PAGE 26 ========================
===============

Page Content:
("Metadata: {'producer': 'HarperCollins Publishers', 'creator': 'Harper "
 "Business', 'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "'The_Daily_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
"
 "'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
 "Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
 "'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
 "'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
 "'creationDate': 'D:20040101000000', 'page': 25}")

================================================================================
===============
```

As observed, some pages are empty, and the numbering begins at 1. We will address this during the data cleaning process.

2.2. Parsing Webpages

This lab demonstrates two useful tools for parsing webpages, though many other options are available.

Firecrawl

`Firecrawl` is a high-performance web crawling tool designed for efficient data extraction and parsing. Learn more at Firecrawl

In [7]:
```python
# Initialize FireCrawlLoader for Web Scraping
loader = FireCrawlLoader(
    api_key=firecrawl_api_key,
    url="https://hbr.org/2005/01/managing-oneself",
    mode="scrape",   #"scrape" mode extracts full-page content while remo
    params={
        "onlyMainContent": True, #  Extract only the main content of the
        "formats": ["markdown"], #  Extract in markdown format
        'excludeTags': ['img', 'a'] # Exclude the images and links from t
    }
)

# Load and extract webpage content
firecrawl_docs = loader.load()
```

In [8]:
```python
# Check the document object type
print(type(firecrawl_docs))

 # Number of documents extracted; should be 1.
print(len(firecrawl_docs))
```

```
<class 'list'>
1
```

In [9]:
```
# Inspect the first document object, print its type and attributes
print(type(firecrawl_docs[0])) # A document object
print(dir(firecrawl_docs[0]))  # List all attributes and methods
```

```
<class 'langchain_core.documents.base.Document'>
['__abstractmethods__', '__annotations__', '__class__', '__class_getitem_
_', '__class_vars__', '__copy__', '__deepcopy__', '__delattr__', '__dict_
_', '__dir__', '__doc__', '__eq__', '__fields__', '__fields_set__', '__for
mat__', '__ge__', '__get_pydantic_core_schema__', '__get_pydantic_json_sch
ema__', '__getattr__', '__getattribute__', '__getstate__', '__gt__', '__ha
sh__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '_
_module__', '__ne__', '__new__', '__pretty__', '__private_attributes__',
'__pydantic_complete__', '__pydantic_computed_fields__', '__pydantic_core_
schema__', '__pydantic_custom_init__', '__pydantic_decorators__', '__pydan
tic_extra__', '__pydantic_fields__', '__pydantic_fields_set__', '__pydanti
c_generic_metadata__', '__pydantic_init_subclass__', '__pydantic_parent_na
mespace__', '__pydantic_post_init__', '__pydantic_private__', '__pydantic_
root_model__', '__pydantic_serializer__', '__pydantic_validator__', '__red
uce__', '__reduce_ex__', '__replace__', '__repr__', '__repr_args__', '__re
pr_name__', '__repr_recursion__', '__repr_str__', '__rich_repr__', '__seta
ttr__', '__setstate__', '__signature__', '__sizeof__', '__slots__', '__str
__', '__subclasshook__', '__weakref__', '_abc_impl', '_calculate_keys', '_
check_frozen', '_copy_and_set_values', '_get_value', '_iter', 'cast_id_to_
str', 'construct', 'copy', 'dict', 'from_orm', 'get_lc_namespace', 'id',
'is_lc_serializable', 'json', 'lc_attributes', 'lc_id', 'lc_secrets', 'met
adata', 'model_computed_fields', 'model_config', 'model_construct', 'model
_copy', 'model_dump', 'model_dump_json', 'model_extra', 'model_fields', 'm
odel_fields_set', 'model_json_schema', 'model_parametrized_name', 'model_p
ost_init', 'model_rebuild', 'model_validate', 'model_validate_json', 'mode
l_validate_strings', 'page_content', 'parse_file', 'parse_obj', 'parse_ra
w', 'schema', 'schema_json', 'to_json', 'to_json_not_implemented', 'type',
'update_forward_refs', 'validate']
```

In [10]:
```
# Print the metadata of the first document
for key, value in firecrawl_docs[0].metadata.items():
    print(f"{key}: {value}")
```

```
twitter:site: @harvardbiz
test: ['rollback', 'rollback']
favicon: https://hbr.org/resources/images/android-chrome-192x192.png
msapplication-config: none
robots: max-image-preview:large
uid: 28FA1DC59350BA3EDC440C9249BDD7BE
viewport: ['width=device-width', 'initial-scale=1, width=device-width', 'w
idth=device-width']
page-type: ITEM
next-head-count: 14
language: en
twitter:card: summary_large_image
article-type: premium
ox-group: 537063787
title: Managing Oneself
twitter:creator: @harvardbiz
scrapeId: d5dae20d-1fcd-485b-9c5f-d937bf56e934
sourceURL: https://hbr.org/2005/01/managing-oneself
url: https://hbr.org/2005/01/managing-oneself
statusCode: 200
```

In [11]:
```
# Display the extracted webpage content
display(Markdown(firecrawl_docs[0].page_content))
```

Managing Oneself

Success in the knowledge economy comes to those who know themselves—their strengths, their values, and how they best perform.by

isitsharp/Getty Images

Summary.

Throughout history, people had little need to manage their careers—they were born into their stations in life or, in the recent past, they relied on their companies to chart their career paths. But times have drastically changed. Today we must all...more

- Post
- Share
- Save
- Print

History's great achievers—a Napoléon, a da Vinci, a Mozart—have always managed themselves. That, in large measure, is what makes them great achievers. But they are rare exceptions, so unusual both in their talents and their accomplishments as to be considered outside the boundaries of ordinary human existence. Now, most of us, even those of us with modest endowments, . We will have to learn to develop ourselves. We will have to place ourselves where we can make the greatest contribution. And we will have to stay mentally alert and engaged during a 50-year working life, which means knowing how and when to change the work we do.

What Are My Strengths?

Most people think they know what they are good at. They are usually wrong. More often, people know what they are not good at—and even then more people are wrong than right. And yet, a person can perform only from strength. One cannot build performance on weaknesses, let alone on something one cannot do at all.

Throughout history, people had little need to know their strengths. A person was born into a position and a line of work: The peasant's son would also be a peasant; the artisan's daughter, an artisan's wife; and so on. But now people have choices. We in order to know where we belong.

The only way to discover your strengths is through feedback analysis. Whenever you make a key decision or take a key action, write down what you expect will happen. Nine or 12 months later, compare the actual results with your expectations. I have been practicing this method for 15 to 20 years now, and every time I do it, I am surprised. The feedback analysis showed me, for instance—and to my great surprise—that I have an intuitive understanding of technical people, whether they are engineers or accountants earchers. It also showed me that I don't really resonate with generalists.

. It was invented sometime in the 14th century by an otherwise totally obscure German theologian and picked up quite independently, some 150 years later, by John Calvin and Ignatius of Loyola, each of who incorporated it into the practice of his followers. In fact, the steadfast focus on performance and results that this habit produces explains why the institutions these two men founded, the Calvinist church and the Jesuit order, came to dominate Europe within 30 years.

Practiced consistently, this simple method will show you within a fairly short period of time, maybe two or three years, where your strengths lie—and this is the most important thing to know. The method will show you what you are doing or failing to do that deprives you of the full benefits of your strengths. It will show you where you are not particularly competent. And finally, it will show you where you have no strengths and cannot perform.

Several implications for action follow from feedback analysis. First and foremost, concentrate on your strengths. Put yourself where your strengths can produce results.

Second, work on improving your strengths. Analysis will rapidly show where you need to improve skills or acquire new ones. It will also show the gaps in your knowledge—and those can usually be filled. Mathematicians are born, but everyone can learn trigonometry.

Third, discover where your intellectual arrogance is causing disabling ignorance and overcome it. Far too many people—especially people with great expertise in one area—are contemptuous of knowledge in other areas or believe that being bright is a substitute for knowledge. First-rate engineers, for instance, tend to take pride in not knowing anything about people. Human beings, they believe, are much too disorderly for the good engineering mind. Human resources professionals, by contrast, often pride themselves on their ignorance of elementary accounting or of quantitative methods altogether. But taking pride in such ignorance is self-defeating. Go to work on acquiring the skills and knowledge you need to fully realize your strengths.

It takes far more energy to improve from incompetence to mediocrity than to improve from first-rate performance to excellence.

It is equally essential to —the things you do or fail to do that inhibit your effectiveness and performance. Such habits will quickly show up in the feedback. For example, a planner may find that his beautiful plans fail because he does not follow through on them. Like so many brilliant people, he believes that ideas move mountains. But bulldozers move mountains; ideas show where the bulldozers should go to work. This planner will have to learn that the work does not stop when the plan is completed. He must find people to carry out the plan and explain it to them. He must adapt and change it as he puts it into action. And finally, he must decide when to stop pushing the plan.

At the same time, feedback will also reveal when the problem is a lack of manners. Manners are the lubricating oil of an organization. It is a law of nature that two moving bodies in contact with each other create friction. This is as true for human beings as it is for inanimate objects. Manners—simple things like saying “please” and “thank you” and

knowing a person's name or asking after her family—enable two people to work together whether they like each other or not. Bright people, especially bright young people, often do not understand this. If analysis shows that someone's brilliant work fails again and again as soon as cooperation from others is required, it probably indicates a lack of courtesy—that is, a lack of manners.

Comparing your expectations with your results also indicates what not to do. We all have a vast number of areas in which we have no talent or skill and little chance of becoming even mediocre. In those areas a person—and especially a knowledge worker—should not take on work, jobs, and assignments. One should waste as little effort as possible on improving areas of low competence. It takes far more energy and work to improve from incompetence to mediocrity than it takes to improve from to excellence. And yet most people—especially most teachers and most organizations—concentrate on making incompetent performers into mediocre ones. Energy, resources, and time should go instead to making a competent person into a star performer.

How Do I Perform?

Amazingly few people know how they get things done. Indeed, most of us do not even know that different people work and perform differently. Too many people work in ways that are not their ways, and that almost guarantees nonperformance. For knowledge workers, How do I perform? may be an even more important question than What are my strengths?

Like one's strengths, how one performs is unique. It is a matter of personality. Whether personality be a matter of nature or nurture, it surely is formed long before a person goes to work. And *how* a person performs is a given, just as *what* a person is good at or not good at is a given. A person's way of performing can be slightly modified, but it is unlikely to be completely changed—and certainly not easily. Just as people achieve results by doing what they are good at, they also achieve results by working in ways that they best perform. A few common personality traits usually determine how a person performs.

Am I a reader or a listener?

The first thing to know is whether you are a reader or a listener. Far too few people even know that there are readers and listeners and that people are rarely both. Even fewer know which of the two they themselves are. But some examples will show how damaging such ignorance can be.

When Dwight Eisenhower was Supreme Commander of the Allied forces in Europe, he was the darling of the press. His press conferences were famous for their style—General Eisenhower showed total command of whatever question he was asked, and he was able to describe a situation and explain a policy in two or three beautifully polished and elegant sentences. Ten years later, the same journalists who had been his admirers held President Eisenhower in open contempt. He never addressed the questions, they ut rambled on endlessly about something else. And they constantly

ridiculed him for butchering the King's English in incoherent and ungrammatical answers.

Read more about

Eisenhower apparently did not know that he was a reader, not a listener. When he was Supreme Commander in Europe, his aides made sure that every question from the press was presented in writing at least half an hour before a conference was to begin. And then Eisenhower was in total command. When he became president, he succeeded two listeners, Franklin D. Roosevelt and Harry Truman. and both enjoyed free-for-all press conferences. Eisenhower may have felt that he had to do what his two predecessors had done. As a result, he never even heard the questions journalists asked. And Eisenhower is not even an extreme case of a nonlistener.

A few years later, Lyndon Johnson destroyed his presidency, in large measure, by not knowing that he was a listener. His predecessor, John Kennedy, was a reader who had assembled a brilliant group of writers as his assistants, making sure that they wrote to him before discussing their memos in person. Johnson kept these people on his staff—and they kept on writing. He never, apparently, understood one word of what they wrote. Yet as a senator, Johnson had been superb; for parliamentarians have to be, above all, listeners.

Few listeners can be made, or can make themselves, into competent readers—and vice versa. The listener who tries to be a reader will, therefore, suffer the fate of Lyndon Johnson, whereas the reader who tries to be a listener will suffer the fate of Dwight Eisenhower. They will not perform or achieve.

How do I learn?

The second thing to know about how one performs is to know how one learns. Many first-class writers—Winston Churchill is but one example—do poorly in school. They tend to remember their schooling as pure torture. Yet few of their classmates remember it the same way. They may not have enjoyed the school very much, but the worst they suffered was boredom. The explanation is that writers do not, as a rule, learn by listening and reading. They learn by writing. Because schools do not allow them to learn this way, they get poor grades.

Schools everywhere are organized on the and that it is the same way for everybody. But to be forced to learn the way a school teaches is sheer hell for students who learn differently. Indeed, there are probably half a dozen different ways to learn.

There are people, like Churchill, who learn by writing. Some people learn by taking copious notes. Beethoven, for example, left behind an enormous number of sketchbooks, yet he said he never actually looked at them when he composed. Asked why he kept them, he is reported to have replied, "If I don't write it down immediately, I forget it right away. If I put it into a sketchbook, I never forget it and I never have to look it up again." Some people learn by doing. Others learn by hearing themselves talk.

Further Reading

A chief executive I know who converted a small and mediocre family business into the leading company in its industry was one of those people who learn by talking. He was in the habit of calling his entire senior staff into his office once a week and then talking at them for two or three hours. He would raise policy issues and argue three different positions on each one. He rarely asked his associates for comments or questions; he simply needed an audience to hear himself talk. That's how he learned. And although he is a fairly extreme case, learning through talking is by no means an unusual method. Successful trial lawyers learn the same way, as do many medical diagnosticians (and so do I).

Of all the important pieces of self-knowledge, understanding how you learn is the easiest to acquire. When I ask people, "How do you learn?" most of them know the answer. But when I ask, "Do you act on this knowledge?" few answer yes. And yet, acting on this knowledge is the key to performance; or rather, *not* acting on this knowledge condemns one to nonperformance.

Am I a reader or a listener? and How do I learn? are the first questions to ask. But they are by no means the only ones. To manage yourself effectively, you also have to ask, Do I work well with people, or am I a loner? And if you do work well with people, you then must ask, In what relationship?

Some people work best as subordinates. General George Patton, the great American military hero of World War II, is a prime example. Patton was America's top troop commander. Yet when he was proposed for an independent command, General George Marshall, the U.S. chief of staff—and probably the most successful picker of men in U.S. history—said, "Patton is the best subordinate the American army has ever produced, but he would be the worst commander."

Some people work best as team members. Others work best alone. Some are exceptionally talented as coaches and mentors; others are simply incompetent as mentors.

Do not try to change yourself—you are unlikely to succeed. Work to improve the way you perform.

Another crucial question is, Do I produce results as a decision-maker or as an adviser? A great many people perform best as advisers but cannot take the burden and pressure of making the decision. A good many other people, by contrast, need an adviser to force themselves to think; then they can make decisions and act on them with speed, self-confidence, and courage.

This is a reason, by the way, that the number two person in an organization often fails when promoted to the number one position. The top spot requires a decision-maker. Strong decision-makers often put somebody they trust into the number two spot as their adviser—and in that position the person is outstanding. But in the number one spot, the same person fails. He or she knows what the decision should be but cannot [illegible]ponsibility of actually making it.

Other important questions to ask include, , or do I need a highly structured and predictable environment? Do I work best in a big organization or a small one? Few people work well in all kinds of environments. Again and again, I have seen people who were very successful in large organizations flounder miserably when they moved into smaller ones. And the reverse is equally true.

The conclusion bears repeating: Do not try to change yourself—you are unlikely to succeed. But work hard to improve the way you perform. And try not to take on work you cannot perform or will only perform poorly.

What Are My Values?

To be able to manage yourself, you finally have to ask, What are my values? This is not a question of ethics. With respect to ethics, the rules are the same for everybody, and the test is a simple one. I call it the "mirror test."

In the early years of this century, the most highly respected diplomat of all the great powers was the German ambassador in London. He was clearly destined for great things—to become his country's foreign minister, at least, if not its federal chancellor. Yet in 1906 he abruptly resigned rather than preside over a dinner given by the diplomatic corps for Edward VII. The king was a notorious womanizer and made it clear what kind of dinner he wanted. The ambassador is reported to have said, "I refuse to see a pimp in the mirror in the morning when I shave."

That is the mirror test. that you ask yourself, What kind of person do I want to see in the mirror in the morning? What is ethical behavior in one kind of organization or situation is ethical behavior in another. But ethics is only part of a value system—especially of an organization's value system.

To work in an organization whose value system is unacceptable or incompatible with one's own condemns a person both to frustration and to nonperformance.

Consider the experience of a highly successful human resources executive whose company was acquired by a bigger organization. After the acquisition, she was promoted to do the kind of work she did best, which included selecting people for important positions. The executive deeply believed that a company should hire people for such positions from the outside only after exhausting all the inside possibilities. But her new company believed in first looking outside "to bring in fresh blood." There is something to be said for both approaches—in my experience, the proper one is to do some of both. They are, however, fundamentally incompatible—not as policies but as values. They bespeak different views of the relationship between organizations and people; different views of the responsibility of an organization to its people and their development; and different views of a person's most important contribution to an enterprise. After several years of frustration, the executive quit—at considerable financial loss. Her values and the values of the organization simply were not compatible.

Similarly, whether a pharmaceutical company tries to obtain results by making constant, small improvements or by achieving occasional, highly expensive, and risky "breakthroughs" is not primarily an economic question. The results of either strategy may be pretty much the same. At bottom, there is a conflict between a value system that sees the company's contribution in terms of helping physicians do better what they already do and a value system that is oriented toward making scientific discoveries.

Whether a business should be run for short-term results or with a focus on the long term is likewise a question of values. Financial analysts believe that businesses can be run for both simultaneously. Successful businesspeople know better. To be sure, every company has to produce short-term results. But in any conflict between short-term results and long-term growth, each company will determine its own priority. This is not primarily a disagreement about economics. It is fundamentally a value conflict regarding the function of a business and the responsibility of management.

. One of the fastest-growing pastoral churches in the United States measures success by the number of new parishioners. Its leadership believes that what matters is how many newcomers join the congregation. The Good Lord will then minister to their spiritual needs or at least to the needs of a sufficient percentage. Another pastoral, evangelical church believes that what matters is people's spiritual growth. The church eases out newcomers who join but do not enter into its spiritual life.

Again, this is not a matter of numbers. At first glance, it appears that the second church grows more slowly. But it retains a far larger proportion of newcomers than the first one does. Its growth, in other words, is more solid. This is also not a theological problem, or only secondarily so. It is a problem about values. In a public debate, one pastor argued, "Unless you first come to church, you will never find the gate to the Kingdom of Heaven."

"No," answered the other. "Until you first look for the gate to the Kingdom of Heaven, you don't belong in church."

Organizations, like people, have values. To be effective in an organization, a person's values must be compatible with the organization's values. They do not need to be the same, but they must be close enough to coexist. Otherwise, the person will not only be frustrated but also will not produce results.

A person's strengths and the way that person performs rarely conflict; the two are complementary. But there is sometimes a conflict between a person's values and his or her strengths. What one does well—even very well and successfully—may not fit with one's value system. In that case, the work may not appear to be worth devoting one's life to (or even a substantial portion thereof).

If I may, allow me to interject a personal note. Many years ago, I too had to decide between my values and what I was doing successfully. I was doing very well as a young investment banker in London in the mid-1930s, and the work clearly fit my strengths. Yet I did not see myself making a contribution as an asset manager. People, I realized, were what I valued, and I saw no point in being the richest man in the cemetery. I had no

money and no other job prospects. Despite the continuing Depression, I quit—and it was the right thing to do. Values, in other words, are and should be the ultimate test.

Where Do I Belong?

A small number of people know very early where they belong. Mathematicians, musicians, and cooks, for instance, are usually mathematicians, musicians, and cooks by the time they are four or five years old. Physicians usually decide on their careers in their teens, if not earlier. But most people, especially highly gifted people, do not really know where they belong until they are well past their mid-twenties. By that time, however, they should know the answers to the three questions: What are my strengths? How do I perform? and, What are my values? And then they can and should decide where they belong.

Or rather, they should be able to decide where they do *not* belong. The person who has learned that he or she does not perform well in a big organization should have learned to say no to a position in one. The person who has learned that he or she is not a should have learned to say no to a decision-making assignment. A General Patton (who probably never learned this himself) should have learned to say no to an independent command.

Equally important, knowing the answer to these questions enables a person to say to an opportunity, an offer, or an assignment, “Yes, I will do that. But this is the way I should be doing it. This is the way it should be structured. This is the way the relationships should be. These are the kind of results you should expect from me, and in this time frame, because this is who I am.”

Successful careers are not planned. They develop when people are prepared for opportunities because they know their strengths, their method of work, and their values. Knowing where one belongs can transform an ordinary person—hardworking and competent but otherwise mediocre—into an outstanding performer.

What Should I Contribute?

Throughout history, the great majority of people never had to ask the question, ? They were told what to contribute, and their tasks were dictated either by the work itself—as it was for the peasant or artisan—or by a master or a mistress—as it was for domestic servants. And until very recently, it was taken for granted that most people were subordinates who did as they were told. Even in the 1950s and 1960s, the new knowledge workers (the so-called organization men) looked to their company’s personnel department to plan their careers.

Then in the late 1960s, no one wanted to be told what to do any longer. Young men and women began to ask, What do *I* want to do? And what they heard was that the way to contribute was to “do your own thing.” But this solution was as wrong as the

organization men's had been. Very few of the people who believed that doing one's own thing would lead to contribution, self-fulfillment, and success achieved any of the three.

But still, there is no return to the old answer of doing what you are told or assigned to do. Knowledge workers in particular have to learn to ask a question that has not been asked before: What *should* my contribution be? To answer it, they must address three distinct elements: What does the situation require? Given my strengths, my way of performing, and my values, how can I make the greatest contribution to what needs to be done? And finally, What results have to be achieved to make a difference?

Consider the experience of a newly appointed hospital administrator. The hospital was big and prestigious, but it had been coasting on its reputation for 30 years. The new administrator decided that his contribution should be to establish a standard of excellence in one important area within two years. He chose to focus on the emergency room, which was big, visible, and sloppy. He decided that every patient who came into the ER had to be seen by a qualified nurse within 60 seconds. Within 12 months, the hospital's emergency room had become a model for all hospitals in the United States, and within another two years, the whole hospital had been transformed.

As this example suggests, it is rarely possible—or even particularly fruitful—to look too far ahead. A plan can usually cover no more than 18 months and still be reasonably clear and specific. So the question in most cases should be, Where and how can I achieve results that will make a difference within the next year and a half? The answer must balance several things. First, the results should be hard to achieve—they should require "stretching," to use the current buzzword. But also, they should be within reach. To aim at results that cannot be achieved—or that can be only under the most unlikely circumstances—is not being ambitious; it is being foolish. Second, the results should be meaningful. They should make a difference. Finally, results should be visible and, if at all possible, measurable. From this will come a course of action: what to do, where and how to start, and what goals and deadlines to set.

Responsibility for Relationships

Very few people work by themselves and achieve results by themselves—a few great artists, a few great scientists, a few great athletes. Most people work with others and are effective with other people. That is true whether they are members of an organization or independently employed. Managing yourself requires taking responsibility for relationships. This has two parts.

The first is to accept the fact that other people are as much individuals as you yourself are. They perversely insist on behaving like human beings. This means that they too have their strengths; they too have their ways of getting things done; they too have their values. To be effective, therefore, you have to know the strengths, the performance modes, and the values of your coworkers.

That sounds obvious, but few people pay attention to it. Typical is the person who was trained to write reports in his or her first assignment because that boss was a reader.

Even if the next boss is a listener, the person goes on writing reports that, invariably, produce no results. Invariably the boss will think the employee is stupid, incompetent, and lazy, and he or she will fail. But that could have been avoided if the employee had only looked at the new boss and analyzed how *this* boss performs.

Bosses are neither a title on the organization chart nor a "function." They are individuals and are entitled to do their work in the way they do it best. It is incumbent on the people who work with them to observe them, to find out how they work, and to adapt themselves to what makes their bosses most effective. This, in fact, is the secret of "managing" the boss.

The same holds true for all your coworkers. Each works his or her way, not your way. And each is entitled to work in his or her way. What matters is whether they perform and what their values are. As for how they perform—each is likely to do it differently. The first secret of effectiveness is to understand the people you work with and depend on so that you can make use of their strengths, their ways of working, and their values. Working relationships are as much based on the people as they are on the work.

The second part of relationship responsibility is taking responsibility for communication. Whenever I, or any other consultant, start to work with an organization, the first thing I hear about are all the personality conflicts. Most of these arise from the fact that people do not know what other people are doing and how they do their work, or what contribution the other people are concentrating on and what results they expect. And the reason they do not know is that they have not asked and therefore have not been told.

This failure to ask reflects human stupidity less than it reflects human history. Until recently, it was unnecessary to tell any of these things to anybody. In the medieval city, everyone in a district plied the same trade. In the countryside, everyone in a valley planted the same crop as soon as the frost was out of the ground. Even those few people who did things that were not "common" worked alone, so they did not have to tell anyone what they were doing.

Today the great majority of people work with others who have different tasks and responsibilities. The marketing vice president may have come out of sales and know everything about sales, but she knows nothing about the things she has never done—pricing, advertising, packaging, and the like. So the people who do these things must make sure that the marketing vice president understands what they are trying to do, why they are trying to do it, how they are going to do it, and what results to expect.

If the marketing vice president does not understand what these high-grade knowledge specialists are doing, it is primarily their fault, not hers. They have not educated her. Conversely, it is the marketing vice president's responsibility to make sure that all of her coworkers understand how she looks at marketing: what her goals are, how she works, and what she expects of herself and of each one of them.

The first secret of effectiveness is to understand the people you work with so that you can make use of their strengths.

Even people who understand the importance of often do not communicate sufficiently with their associates. They are afraid of being thought presumptuous or inquisitive or stupid. They are wrong. Whenever someone goes to his or her associates and says, "This is what I am good at. This is how I work. These are my values. This is the contribution I plan to concentrate on and the results I should be expected to deliver," the response is always, "This is most helpful. But why didn't you tell me earlier?"

And one gets the same reaction—without exception, in my experience—if one continues by asking, "And what do I need to know about your strengths, how you perform, your values, and your proposed contribution?" In fact, knowledge workers should request this of everyone with whom they work, whether as subordinate, superior, colleague, or team member. And again, whenever this is done, the reaction is always, "Thanks for asking me. But why didn't you ask me earlier?"

Organizations are no longer built on force but on trust. The existence of trust between people does not necessarily mean that they like one another. It means that they understand one another. Taking responsibility for relationships is therefore an absolute necessity. It is a duty. Whether one is a member of the organization, a consultant to it, a supplier, or a distributor, one owes that responsibility to all one's coworkers: those whose work one depends on as well as those who depend on one's own work.

The Second Half of Your Life

When work for most people meant manual labor, there was no need to worry about the second half of your life. You simply kept on doing what you had always done. And if you were lucky enough to survive 40 years of hard work in the mill or on the railroad, you were quite happy to spend the rest of your life doing nothing. Today, however, most work is knowledge work, and knowledge workers are not "finished" after 40 years on the job, they are merely bored.

We hear a great deal of talk about the midlife crisis of the executive. It is mostly boredom. At 45, most executives have reached the peak of their business careers, and they know it. After 20 years of doing very much the same kind of work, they are very good at their jobs. But they are not learning or contributing or deriving challenge and satisfaction from the job. And yet they are still likely to face another 20 if not 25 years of work. That is why managing oneself increasingly leads one to begin a second career.

There are three ways to develop a second career. The first is actually to start one. Often this takes nothing more than moving from one kind of organization to another: the divisional controller in a large corporation, for instance, becomes the controller of a medium-sized hospital. But there are also growing numbers of people who move into different lines of work altogether: the business executive or government official who enters the ministry at 45, for instance; or the midlevel manager who leaves corporate life after 20 years to attend law school and become a small-town attorney.

undertaken by people who have achieved modest success in their first jobs. Such people have substantial skills, and they know how to work. They need a community—the house is empty with the children gone—and they need income as well. But above all, they need challenge.

The second way to prepare for the second half of your life is to develop a parallel career. Many people who are very successful in their first careers stay in the work they have been doing, either on a full-time or part-time or consulting basis. But in addition, they create a parallel job, usually in a nonprofit organization, that takes another 10 hours of work a week. They might take over the administration of their church, for instance, or the presidency of the local Girl Scouts council. They might run the battered women's shelter, work as a children's librarian for the local public library, sit on the school board, and so on.

Finally, there are the social entrepreneurs. These are usually people who have been very successful in their first careers. They love their work, but it no longer challenges them. In many cases they keep on doing what they have been doing all along but spend less and less of their time on it. They also start another activity, usually a nonprofit. My friend Bob Buford, for example, built a very successful television company that he still runs. But he has also founded and built a successful nonprofit organization that works with Protestant churches, and he is building another to teach social entrepreneurs how to manage their own nonprofit ventures while still running their original businesses.

People who manage the second half of their lives may always be a minority. The majority may "retire on the job" and count the years until their actual retirement. But it is this minority, the men and women who see a long working-life expectancy as an opportunity both for themselves and for society, who will become leaders and models.

There is one prerequisite for managing the second half of your life: You must begin long before you enter it. When it first became clear 30 years ago that working-life expectancies were lengthening very fast, many observers (including myself) believed that retired people would increasingly become volunteers for nonprofit institutions. That has not happened. If one does not begin to volunteer before one is 40 or so, one will not volunteer once past 60.

Similarly, all the I know began to work in their chosen second enterprise long before they reached their peak in their original business. Consider the example of a successful lawyer, the legal counsel to a large corporation, who has started a venture to establish model schools in his state. He began to do volunteer legal work for the schools when he 5. He was elected to the school board at age 40. At age 50, when he had

amassed a fortune, he started his own enterprise to build and to run model schools. He is, however, still working nearly full-time as the lead counsel in the company he helped found as a young lawyer.

There is one prerequisite for managing the second half of your life: You must begin doing so long before you enter it.

There is another reason to develop a second major interest, and to develop it early. No one can expect to live very long without experiencing a serious setback in his or her life or work. There is the competent engineer who is passed over for promotion at age 45. There is the competent college professor who realizes at age 42 that she will never get a professorship at a big university, even though she may be fully qualified for it. There are tragedies in one's family life: the breakup of one's marriage or the loss of a child. At such times, a second major interest—not just a hobby—may make all the difference. The engineer, for example, now knows that he has not been very successful in his job. But in his outside activity—as church treasurer, for example—he is a success. One's family may break up, but in that outside activity there is still a community.

In a society in which success has become so terribly important, having options will become increasingly vital. Historically, there was no such thing as "success." The overwhelming majority of people did not expect anything but to stay in their "proper station," as an old English prayer has it. The only mobility was downward mobility.

In a knowledge society, however, we expect everyone to be a success. This is clearly an impossibility. For a great many people, there is at best an absence of failure. Wherever there is success, there has to be failure. And then it is vitally important for the individual, and equally for the individual's family, to have an area in which he or she can contribute, make a difference, and be *somebody.* That means finding a second area—whether in a second career, a parallel career, or a social venture—that offers an opportunity for being a leader, for being respected, for being a success.

The may seem obvious, if not elementary. And the answers may seem self-evident to the point of appearing naive. But managing oneself requires new and unprecedented things from the individual, and especially from the knowledge worker. In effect, managing oneself demands that each knowledge worker think and behave like a chief executive officer. Further, the shift from manual workers who do as they are told to knowledge workers who have to manage themselves profoundly challenges social structure. Every existing society, even the most individualistic one, takes two things for granted, if only subconsciously: that organizations outlive workers, and that most people stay put.

But today the opposite is true. Knowledge workers outlive organizations, and they are mobile. The need to manage oneself is therefore creating a revolution in human affairs.

This article is also included in the book (Harvard Business Review Press, 2022).

Read more on or related topics , and

A version of this article appeared in the issue of *Harvard Business Review.*

- Post
- Share
- Save
- Print

Read more on or related topics , and

iframe

Recommended For You

Partner Center

LlamaParse

`LlamaParse` is a robust document parsing tool by LlamaIndex, providing high-quality extraction for complex webpages. Learn more at LlamaParse.

In [12]:
```
# Import LlamaParse for web scraping and data extraction
from llama_cloud_services import LlamaParse

# Import nest_asyncio to handle nested asynchronous event loops
import nest_asyncio

# Required for running in Jupyter Notebook
nest_asyncio.apply() # # Allows running async code inside a Jupyter Noteb

# Define the URL of the webpage to be scraped
url ="https://hbr.org/2005/01/managing-oneself"

# Initialize the web scraper using LlamaParse
web_scrapper = LlamaParse(api_key=llama_cloud_api_key, # API Key
                          result_type="markdown",      # Specify the outp
                          balance_mode=True,           # Enable balanced
                          extract_metadata=True,       # Explicitly enabl
                          page_separator="\n",         # Define how pages
                          split_by_page = False,       # Whether to split
                          veribose =True)              # Enable detailed

# Load and extract data from the webpage
# it returns a list of document objects, where each document contains tex
web_docs = web_scrapper.load_data(url)
```

```
Started parsing the file under job_id 2a0e1b0b-ec62-4b6c-a904-f5d9b252e17d
```

In [13]:
```
# check the document object type
print(type(web_docs))

 # Number of documents extracted; should be 1.
print(len(web_docs))
```

```
<class 'list'>
1
```

In [14]:
```
# inspect the first document object, print its type and attributes
print(type(web_docs[0])) # A document object
print(dir(web_docs[0]))  # List all attributes and methods
```

```
<class 'llama_index.core.schema.Document'>
['__abstractmethods__', '__annotations__', '__class__', '__class_getitem_
_', '__class_vars__', '__copy__', '__deepcopy__', '__delattr__', '__dict_
_', '__dir__', '__doc__', '__eq__', '__fields__', '__fields_set__', '__for
mat__', '__ge__', '__get_pydantic_core_schema__', '__get_pydantic_json_sch
ema__', '__getattr__', '__getattribute__', '__getstate__', '__gt__', '__ha
sh__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '_
_module__', '__ne__', '__new__', '__pretty__', '__private_attributes__',
'__pydantic_complete__', '__pydantic_computed_fields__', '__pydantic_core_
schema__', '__pydantic_custom_init__', '__pydantic_decorators__', '__pydan
tic_extra__', '__pydantic_fields__', '__pydantic_fields_set__', '__pydanti
c_generic_metadata__', '__pydantic_init_subclass__', '__pydantic_parent_na
mespace__', '__pydantic_post_init__', '__pydantic_private__', '__pydantic_
root_model__', '__pydantic_serializer__', '__pydantic_validator__', '__red
uce__', '__reduce_ex__', '__replace__', '__repr__', '__repr_args__', '__re
pr_name__', '__repr_recursion__', '__repr_str__', '__rich_repr__', '__seta
ttr__', '__setstate__', '__signature__', '__sizeof__', '__slots__', '__str
__', '__subclasshook__', '__weakref__', '_abc_impl', '_calculate_keys', '_
check_frozen', '_copy_and_set_values', '_get_value', '_iter', 'as_related_
node_info', 'audio_resource', 'child_nodes', 'class_name', 'construct', 'c
opy', 'custom_model_dump', 'dict', 'doc_id', 'embedding', 'example', 'excl
uded_embed_metadata_keys', 'excluded_llm_metadata_keys', 'extra_info', 'fr
om_cloud_document', 'from_dict', 'from_embedchain_format', 'from_haystack_
format', 'from_json', 'from_langchain_format', 'from_orm', 'from_semantic_
kernel_format', 'get_content', 'get_doc_id', 'get_embedding', 'get_metadat
a_str', 'get_type', 'hash', 'id_', 'image_resource', 'json', 'metadata',
'metadata_separator', 'metadata_template', 'model_computed_fields', 'model
_config', 'model_construct', 'model_copy', 'model_dump', 'model_dump_jso
n', 'model_extra', 'model_fields', 'model_fields_set', 'model_json_schem
a', 'model_parametrized_name', 'model_post_init', 'model_rebuild', 'model_
validate', 'model_validate_json', 'model_validate_strings', 'next_node',
'node_id', 'parent_node', 'parse_file', 'parse_obj', 'parse_raw', 'prev_no
de', 'ref_doc_id', 'relationships', 'schema', 'schema_json', 'set_conten
t', 'source_node', 'text', 'text_resource', 'text_template', 'to_cloud_doc
ument', 'to_dict', 'to_embedchain_format', 'to_haystack_format', 'to_jso
n', 'to_langchain_format', 'to_semantic_kernel_format', 'to_vectorflow',
'update_forward_refs', 'validate', 'video_resource']
```

In [15]:
```
# Explore the first document
display(Markdown(web_docs[0].text[:1000]))  # Display first 1000 characte

# You can remove [:1000] to display the entire content.
```

Managing Yourself

Managing Oneself

by Peter F. Drucker

From the Magazine (January 2005)

Summary. Throughout history, people had little need to manage their careers—they were born into their stations in life or, in the recent past, they relied on their companies to chart their career paths. But times have drastically changed. Today...

History's great achievers—a Napoléon, a da Vinci, a Mozart—have always managed themselves. That, in large measure, is what makes them great achievers. But they are rare exceptions, so unusual both in their talents and their accomplishments as to be considered outside the boundaries of ordinary human existence.

Now, most of us, even those of us with modest endowments, will have to learn to manage ourselves. We will have to learn to develop ourselves. We will have to place ourselves where we can make the greatest contribution. And we will have to stay mentally... alert and engaged during a 50-year working life, which means knowing how and when to c

In [16]:
```
# Explore metadata
print(web_docs[0].metadata)

# Returns an empty {}, meaning LlamaParse did not extract any metadata fr
```

```
{}
```

3. Text cleaning

This step applies basic text cleaning to ensure high-quality data optimized for later retrieval and analysis.

3.1. PDF Cleaning

Common PDF cleaning steps to consider:

- Remove Short/Empty Pages: Filter out pages with insufficient content.
- Remove Duplicates: Identify and remove dupliated pages.
- Remove Irrelevant Content: discard images, tables, watermarks, footers, headers, line breaks, escape characters, etc.
- Handle OCR Noise: If PDFs are scanned documents, applying Optical Character Recognition (OCR) error correction to fix misread characters.

This code below is a simple PDF cleaning function that filters out short pages and duplicate content.It also tracks cleaning statistics.

In [17]:
```python
# Simple PDF Document Cleaning

def clean_pdf_documents(docs, min_content_length=20, verbose=True):
    """
    Cleans a list of PDF document objects by removing:
    - Pages with very short content (e.g., subheaders, blank pages)
    - Duplicate pages based on exact text content

    Parameters:
        - docs (list): A list of document objects, each with `page_conten
        - min_content_length (int): Minimum number of characters required
        - verbose (bool): If True, prints detailed logs of the cleaning p

    Returns:
        tuple: (cleaned_docs, stats)
            - cleaned_docs (list): A list of cleaned document objects.
            - stats (dict): A dictionary with counts of total, removed, a
    """

    stats = {"total": len(docs), "short": 0, "duplicate": 0}

    # Dictionary to store unique content for duplicate detection
    content_map = {}

    # List to store cleaned documents
    cleaned_docs = []

    # Iterate through all document pages
    for i, doc in enumerate(docs):

        # Assign proper page numbering in metadata
        doc.metadata["page"] = i + 1

        # Get the page content and remove unnecessary whitespace
        content = doc.page_content.strip()

        # Check if content is too short (likely irrelevant)
        if len(content) < min_content_length:
            stats["short"] += 1     # count short pages
            if verbose:
                print(f"Skipping page {i+1}: Short content ({len(content)
            continue    # Skip this page

        # Check for duplicates
        if content in content_map:
            stats["duplicate"] += 1     # Count duplicate pages
            if verbose:
                print(f"Skipping page {i+1}: Duplicate of page {content_m
            continue    # Skip this page

        # Add unique content to tracking dictionary (avoid future duplica
        content_map[content] = i

        # Append the valid document to the cleaned list
        cleaned_docs.append(doc)
```

```
    # Calculate the number of valid pages after cleaning
    stats["valid"] = stats["total"] - stats["short"] - stats["duplicate"]

    # Print a cleaning summary if verbose mode is enabled
    if verbose:
        print(f"\nCleaning Summary:")
        print(f"  Total pages: {stats['total']}")
        print(f"  Removed: {stats['short']} short pages, {stats['duplicat
        print(f"  Remaining: {stats['valid']} valid pages")

    # Return cleaned documents and summary statistics
    return cleaned_docs, stats
```

In [18]:

```
# Clean the PDF documents
# Remove pages that are 20 characters or less
cleaned_docs, cleaning_stats = clean_pdf_documents(docs, min_content_leng

# Check if any documents remain after cleaning
if cleaned_docs:
    # Display a sample preview of the first 2 cleaned documents
    for i, doc in enumerate(cleaned_docs[:2]):
        print(f"\n--- Sample Page {i+1} ---")

        # Show a preview of the first 100 characters of page content
        print(f"Content (preview): {doc.page_content[:100]}...")

        # Display associated metadata
        pprint(f"Metadata: {doc.metadata}")
else:
    print("No valid documents remain after cleaning.")
```

```
Skipping page 2: Short content (0 chars)
Skipping page 4: Short content (0 chars)
Skipping page 6: Short content (0 chars)
Skipping page 8: Short content (0 chars)
Skipping page 10: Short content (0 chars)
Skipping page 12: Short content (0 chars)
Skipping page 14: Short content (0 chars)
Skipping page 16: Short content (0 chars)
Skipping page 18: Short content (0 chars)
Skipping page 20: Short content (0 chars)
Skipping page 22: Short content (0 chars)
Skipping page 24: Short content (0 chars)
Skipping page 26: Short content (0 chars)
Skipping page 28: Short content (0 chars)
Skipping page 30: Short content (0 chars)
Skipping page 32: Short content (0 chars)
Skipping page 34: Short content (0 chars)
Skipping page 36: Short content (0 chars)
Skipping page 38: Short content (0 chars)
Skipping page 40: Short content (0 chars)
Skipping page 42: Short content (0 chars)
Skipping page 44: Short content (0 chars)
Skipping page 46: Short content (0 chars)
Skipping page 48: Short content (0 chars)
Skipping page 50: Short content (0 chars)
Skipping page 52: Short content (0 chars)
Skipping page 54: Short content (0 chars)
Skipping page 56: Short content (0 chars)
Skipping page 58: Short content (0 chars)
Skipping page 60: Short content (0 chars)

Cleaning Summary:
  Total pages: 61
  Removed: 30 short pages, 0 duplicates
  Remaining: 31 valid pages

--- Sample Page 1 ---
Content (preview): 1 January
Integrity in Leadership
The spirit of an organization is created from the top.
T he proof ...
("Metadata: {'producer': 'HarperCollins Publishers', 'creator': 'Harper "
 "Business', 'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "'The_Daily_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
"
 "'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
 "Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
 "'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
 "'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
 "'creationDate': 'D:20040101000000', 'page': 1}")

--- Sample Page 2 ---
Content (preview): 2 January
Identifying the Future
The important thing is to identify the “future that has already hap...
("Metadata: {'producer': 'HarperCollins Publishers', 'creator': 'Harper "
 "Business', 'creationdate': 'D:20040101000000', 'source': "
 "'The_Daily_Drucker_January.pdf', 'file_path': "
 "ly_Drucker_January.pdf', 'total_pages': 61, 'format': 'PDF 1.3',
```

```
"
"'title': 'The Daily Drucker (January Excerpt)', 'author': 'Peter F. "
"Drucker', 'subject': 'Management, Leadership, Business', 'keywords': "
"'Management, Leadership, Business, Productivity, Drucker', 'moddate': "
"'D:20250228151450', 'trapped': '', 'modDate': 'D:20250228151450', "
"'creationDate': 'D:20040101000000', 'page': 3}")
```

3.2. Web Data Cleaning

When using other tools for web scraping, we need to convert the extracted content into LangChain's document format to ensure compatibility with our existing pipeline.

This conversion retains both document content and metadata, allowing us to fully utilize LangChain's document processing capabilities.

Below, we demonstrate this process using LlamaParse as an example. We also add metadata to the scrapped content.

In [19]:
```python
# Import LangChain's Document format for compatibility
from langchain_core.documents import Document as LangChainDocument

# Convert LlamaIndex web documents to LangChain format
# Since LlamaParse extracts the entire webpage as a single document,
# we only need to convert the first document (web_docs[0]).
langchain_web_doc = web_docs[0].to_langchain_format()

# Verify that the conversion was successful
# - isinstance() checks if the converted document is now a LangChainDocum
print(f"Conversion successful: {isinstance(langchain_web_doc, LangChainDo

# Display available attributes in the converted document
# - vars() returns all stored attributes inside the object for inspection
print(f"Document attributes: {list(vars(langchain_web_doc).keys())}")
```

```
Conversion successful: True
Document attributes: ['id', 'metadata', 'page_content', 'type']
```

In [20]:
```python
# Import necessary modules for metadata extraction using OpenAI's model
from langchain_community.document_transformers.openai_functions import cr
from langchain_openai import ChatOpenAI

# Define the metadata schema for extraction
schema = {
    "properties": {
        "Title": {"type": "string"},    # Extracts the document title
        "Author": {"type": "string"},   # Extracts the author (if availab
        "Publication": {"type": "string"},  # Extracts the publication so
        "Date": {"type": "string", "description": "Publication date of th
    },
    "required": ["Title"] # Ensures that a title is always included
}

# Initialize the OpenAI model for metadata extraction
llm = ChatOpenAI(temperature=0, model="gpt-4o")  # temperature=0 for dete

# Create a metadata tagger using OpenAI's model
# This function applies the defined schema to extract metadata
tagger = create_metadata_tagger(metadata_schema=schema, llm=llm)
```

```
# Apply metadata tagging to the LangChain-formatted web document
langchain_web_doc_cleaned = document_tagger.transform_documents([langchai

# Display the enhanced document with extracted metadata
print("\nEnhanced document metadata:")

pprint(langchain_web_doc_cleaned[0].metadata)
```

```
Enhanced document metadata:
{'Author': 'Peter F. Drucker',
 'Date': 'January 2005',
 'Publication': 'Harvard Business Review',
 'Title': 'Managing Oneself'}
```

4. Chunking Strategies

Effective text chunking is important for RAG applications, as it directly impacts retrieval accuracy, context relevance, and model efficiency.

This lab explores three primary chunking techniques using the web data:

1. Length-Based Splitting (Baseline Approach)

- Token-based: Splits text by token count, ensuring compatibility with LLMs and reducing truncation issues.
- Character-based: Splits text by character count, providing consistency across different text types.
- Best used for: Fast and simple chunking when semantic continuity is not a primary concern.
- Limitations: May break words or sentences improperly, leading to retrieval of incomplete or disjointed information.

2. Recursive Character Splitting

- Method: Iteratively splits text using a structured hierarchy (paragraphs → sentences → words), preserving coherence.
- Best used for: Retaining readability while maintaining LLM token constraints in RAG pipelines.

3. Semantic Splitting

- Method: Uses embeddings to split text based on meaning rather than arbitrary length limits.
- Benefits: Ensures high recall and precision by keeping semantically related content within the same chunk.
- Best used for: RAG applications requiring accurate, contextually relevant retrieval for better response generation.
- Limitations: Computationally more expensive than length-based methods.

Additional Advanced Splitting Strategies:

- Document-Specific Splitting: Adapts chunking rules based on document types (e.g.,PDFs, Markdown, and source code).
- Agentic Splitting: Uses dynamic chunking logic to optimize retrieval based on query type, cost, or model constraints.
- Alternative Representation Chunking: Converts raw text into structured forms (summaries, embeddings, or keyword graphs) to enhance retrieval efficiency and organization.

Recommended Additional Readings:

- LangChain Text Splitters Documentation
- 5 Levels of Text Splitting - Jupyter Notebook

4.1. Length-Based Splitting

We use Character-Based Splitting as an Example.

- This method divides text into chunks of a fixed character length.
- Ensures chunks stay within processing limits while maintaining readability.

In [21]:
```
from langchain.text_splitter import CharacterTextSplitter

# Initialize the text splitter with defined chunking parameters
text_splitter = CharacterTextSplitter(chunk_size = 512, # Maximum number
                                      chunk_overlap=64, # Overlapping cha
                                      separator="")     # Empty separator

# Apply the splitter to segment the document into manageable chunks
split_docs = text_splitter.split_documents(langchain_web_doc_cleaned)
```

In [22]:
```
# Display a summary of the text splitting results
# - Shows the total number of chunks created after splitting
print(f"Total chunks: {len(split_docs)}\n")

# Compute and display the size of each chunk
chunk_sizes = [len(doc.page_content) for doc in split_docs]
print(" Chunk Sizes:")
print(chunk_sizes)      # Displays a list of chunk lengths for analysis
print("\n" + "-" * 80)  # Separator for readability

# Display first few chunks with formatted output
for i, doc in enumerate(split_docs[:3]):    # Show only the first 3 chunk
    print(f"\n Chunk {i+1}: ({len(doc.page_content)} characters)\n")

    # Format chunk content for better readability
    # - Wraps text at 100 characters per line for clear visualization
    print(textwrap.fill(doc.page_content, width=100))
    print("-" * 80) # Separator for better chunk distinction
```

Total chunks: 90

Chunk Sizes:
[512, 512, 510, 512, 511, 511, 512, 512, 512, 511, 512, 512, 512, 512, 511, 512, 512, 512, 512, 511, 512, 511, 512, 512, 512, 511, 512, 511, 512, 512, 512, 511, 512, 512, 511, 511, 512, 512, 512, 512, 511, 512, 512, 512, 511, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 510, 512, 512, 511, 512, 512, 512, 511, 512, 512, 512, 511, 511, 512, 511, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 511, 512, 512, 512, 512, 212]

Chunk 1: (512 characters)

Managing Yourself # Managing Oneself by Peter F. Drucker From the Magazine (January 2005)
Summary. Throughout history, people had little need to manage their careers—they were born into
their stations in life or, in the recent past, they relied on their companies to chart their career
paths. But times have drastically changed. Today... History’s great achievers—a Napoléon, a da
Vinci, a Mozart—have always managed themselves. That, in large measure, is what makes them great
achievers. But they are rar

Chunk 2: (512 characters)

ge measure, is what makes them great achievers. But they are rare exceptions, so unusual both in
their talents and their accomplishments as to be considered outside the boundaries of ordinary human
existence. Now, most of us, even those of us with modest endowments, will have to learn to manage
ourselves. We will have to learn to develop ourselves. We will have to place ourselves where we can
make the greatest contribution. And we will have to stay mentally... alert and engaged during a
50-year working lif

Chunk 3: (510 characters)

stay mentally... alert and engaged during a 50-year working life, which means knowing how and when
to change the work we do. # What Are My Strengths? Most people think they know what they are good
at. They are usually wrong. More often, people know what they are not good at—and even then more
people are wrong than right. And yet, a person can perform only from strength. One cannot build
performance on weaknesses, let alone on something one cannot do at all. Throughout history, people
had little need to

4.2. Recursive Character Splitting

Recursive character splitting optimizes chunk sizes by first dividing text at natural breakpoints such as paragraphs or sentences. If a chunk exceeds the defined `chunk_size`, it is further split while preserving context.

In LangChain, this process is handled by `RecursiveCharacterTextSplitter`, making it well-suited for long-form documents like research papers and legal texts.

In [23]:
```
# Recursive Splitting
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Initialize RecursiveCharacterTextSplitter
recursive_text_splitter = RecursiveCharacterTextSplitter(chunk_size = 512
                                                         chunk_overlap=64
                                                         )

# Apply recursive splitting to the document
recursive_split_docs = recursive_text_splitter.split_documents(langchain_
```

In [24]:
```
# Display summary of the chunking results
print(f"Total chunks: {len(recursive_split_docs)}\n")

# Compute and display the size of each chunk
chunk_sizes = [len(doc.page_content) for doc in recursive_split_docs]
print(" Chunk Sizes:")
print(chunk_sizes)      # Displays a list of chunk lengths for analysis
print("\n" + "-" * 80)  # Separator for readability

# Display first few chunks with formatted output
for i, doc in enumerate(recursive_split_docs[:3]):    # Show only the fir
    print(f"\n Chunk {i+1}: ({len(doc.page_content)} characters)\n")

    # Format chunk content for better readability
    # - Wraps text at 100 characters per line for clear visualization
    print(textwrap.fill(doc.page_content, width=100))
    print("-" * 80) # Separator for better chunk distinction
```

```
Total chunks: 118

 Chunk Sizes:
[343, 311, 389, 337, 311, 500, 150, 510, 447, 432, 505, 300, 126, 507, 26
2, 507, 292, 243, 503, 74, 365, 504, 173, 312, 502, 224, 508, 224, 511, 7
9, 321, 374, 483, 508, 447, 251, 367, 297, 451, 257, 368, 467, 410, 467, 5
10, 108, 471, 511, 389, 253, 504, 507, 128, 508, 99, 508, 115, 305, 385, 5
04, 160, 20, 511, 161, 449, 385, 333, 511, 110, 421, 503, 431, 232, 510, 4
70, 409, 389, 505, 366, 506, 56, 430, 510, 105, 502, 407, 122, 510, 96, 46
5, 508, 30, 369, 508, 160, 36, 504, 140, 340, 463, 172, 503, 259, 335, 45
5, 249, 407, 121, 510, 408, 335, 505, 147, 501, 270, 468, 448, 120]

--------------------------------------------------------------------------
------

 Chunk 1: (343 characters)

# Managing Yourself  # Managing Oneself  by Peter F. Drucker  From the Mag
azine (January 2005)
Summary. Throughout history, people had little need to manage their career
s—they were born into
their stations in life or, in the recent past, they relied on their compan
ies to chart their career
paths. But times have drastically changed. Today...
--------------------------------------------------------------------------
------

 Chunk 2: (311 characters)

History’s great achievers—a Napoléon, a da Vinci, a Mozart—have always man
aged themselves. That, in
large measure, is what makes them great achievers. But they are rare excep
tions, so unusual both in
their talents and their accomplishments as to be considered outside the bo
undaries of ordinary human
existence.
--------------------------------------------------------------------------
------

 Chunk 3: (389 characters)

Now, most of us, even those of us with modest endowments, will have to lea
rn to manage ourselves. We
will have to learn to develop ourselves. We will have to place ourselves w
here we can make the
greatest contribution. And we will have to stay mentally... alert and enga
ged during a 50-year
working life, which means knowing how and when to change the work we do.
# What Are My Strengths?
--------------------------------------------------------------------------
------
```

RecursiveCharacterTextSplitter prioritizes splitting at natural boundaries like paragraphs and sentences, which can reduce enforced overlap.

Overlap is only applied when splitting at lower levels (e.g., words), so if a clean paragraph break is found, extra overlapping text may not be inserted.

4.3. Semantic Splitting

In LangChain, `SemanticChunker` uses embeddings to split text based on meaning rather than character count or punctuation. It detects semantic breakpoints by analyzing the distance between sequential sentences, often leveraging models from Hugging Face or OpenAI.

To refine chunking decisions, `SemanticChunker` can group sentences (e.g., in sets of 3) before determining split points, reducing noise and improving coherence.

In LangChain, the `breakpoint_threshold_type` parameter controls how semantic breakpoints are detected:

- Percentile (default 95%): Splits where differences exceed a defined percentile.
- Standard Deviation (default 3.0): Splits when differences exceed a set number of standard deviations from the mean.
- Interquartile (default 1.5): Uses the interquartile range to detect split points.
- Gradient (default 95%): Applies anomaly detection on gradients, useful for structured texts like legal or medical documents.

You should also adjust `breakpoint_threshold_amount` to fine-tune sensitivity for more or fewer breakpoints.

In [25]:
```
from langchain_experimental.text_splitter import SemanticChunker

# Initialize the OpenAI embeddings model
embedding_model = OpenAIEmbeddings(model="text-embedding-3-large")

# Initialize the semantic chunker
semantic_text_splitter = SemanticChunker(
    embedding_model,
    breakpoint_threshold_type="percentile", # Determines how breakpoints
    breakpoint_threshold_amount=95  # Sets the percentile threshold for d
)

# Apply semantic splitting to web documents
semantic_split_docs = semantic_text_splitter.split_documents(langchain_we
```

In [26]:
```
# Display summary
print(f"Total semantic chunks: {len(semantic_split_docs)}\n")

# Display chunk sizes
chunk_sizes = [len(doc.page_content) for doc in semantic_split_docs]

print(" Chunk Sizes:")
print(chunk_sizes)
print("\n" + "-" * 80)

# Display first few chunks with formatted output
for i, doc in enumerate(semantic_split_docs[:3]):
    print(f"\n Chunk {i+1}: ({len(doc.page_content)} characters)\n")
    print(textwrap.fill(doc.page_content, width=100))
    print("-" * 80)
```

```
Total semantic chunks: 22

 Chunk Sizes:
[334, 709, 559, 6364, 2247, 230, 2283, 732, 1786, 4182, 1572, 550, 721, 4
0, 6735, 1251, 120, 954, 2171, 4248, 64, 2135]

--------------------------------------------------------------------------
------

 Chunk 1: (334 characters)

# Managing Yourself  # Managing Oneself  by Peter F. Drucker  From the Mag
azine (January 2005)
Summary. Throughout history, people had little need to manage their career
s—they were born into
their stations in life or, in the recent past, they relied on their compan
ies to chart their career
paths. But times have drastically changed.
--------------------------------------------------------------------------
------

 Chunk 2: (709 characters)

Today... History’s great achievers—a Napoléon, a da Vinci, a Mozart—have a
lways managed themselves.
That, in large measure, is what makes them great achievers. But they are r
are exceptions, so unusual
both in their talents and their accomplishments as to be considered outsid
e the boundaries of
ordinary human existence. Now, most of us, even those of us with modest en
dowments, will have to
learn to manage ourselves. We will have to learn to develop ourselves. We
will have to place
ourselves where we can make the greatest contribution. And we will have to
stay mentally... alert
and engaged during a 50-year working life, which means knowing how and whe
n to change the work we
do. # What Are My Strengths?
--------------------------------------------------------------------------
------

 Chunk 3: (559 characters)

Most people think they know what they are good at. They are usually wrong.
More often, people know
what they are not good at—and even then more people are wrong than right.
And yet, a person can
perform only from strength. One cannot build performance on weaknesses, le
t alone on something one
cannot do at all. Throughout history, people had little need to know their
strengths. A person was
born into a position and a line of work: The peasant’s son would also be a
peasant; the artisan’s
daughter, an artisan’s wife; and so on. But now people have choices.
--------------------------------------------------------------------------
------
```

5. Vector store in PostgreSQL

To enable efficient retrieval and analysis, we store both web and book data as vector embeddings in a PostgreSQL vector database.

- Web Data: Processed using semantic chunking.
- Book Data: Each page serves as a natural chunk, maintaining its structured format for accurate analysis.

These documents are then converted into vector embeddings using OpenAI's embedding model and stored in PostgreSQL using PGVector. Metadata is saved in JSONB format, allowing for efficient filtering and retrieval for downstream applications.

Note: we will cover indexing strategies for optimizing vector search in the next lab.

In [27]:
```
from langchain_postgres import PGVector
from sqlalchemy.engine.url import make_url

# Extract the database name from the connection string
database_name = make_url(connection_string).database

# Define document collections to be stored in PostgreSQL
collections = {
    "Web_data": semantic_split_docs,    # Web documents chunked semantica
    "Book_data": cleaned_docs           # Book pages cleaned and used as
}

# Load documents into PostgreSQL vector database
databases = {}
for name, docs in collections.items():
    databases[name] = PGVector.from_documents(
        embedding=embedding_model,      # OpenAI embeddings for vectoriza
        documents=docs,                 # The documents to be stored
        collection_name=name,           # The name of the collection in P
        connection=connection_string,   # Connection string to the Postgr
        use_jsonb=True      # Store metadata as JSONB for efficient query
    )

    # Display confirmation message
    print(f" Successfully loaded {len(docs)} chunks into '{name}' collect
```

```
 Successfully loaded 22 chunks into 'Web_data' collection in 'GenAI_Spring
25_Yongjia_Sun_db'.
 Successfully loaded 31 chunks into 'Book_data' collection in 'GenAI_Sprin
g25_Yongjia_Sun_db'.
```

In [28]:
```
# Check the web_date collection
print(databases["Web_data"])
```

```
<langchain_postgres.vectorstores.PGVector object at 0x118315540>
```

6. Document Retrieval

In this section, we performed a similarity search on both web and book datasets stored in PGVector. The process includes:

- Embedding the Query: Convert the input question into a vector representation.

- Retrieving Top Matches: Searche for the most relevant chunks in both datasets.
- Sorting by Relevance: Sort the results by similarity score, ensuring the most relevant documents appeared first.
- Displaying Results: Format and display the retrieved text chunks for comparison.

In [29]:
```
# Define the search query
query_text = "Why should I focus on my strengths rather than fixing my we

# Perform similarity search on both collections
# - Search for the top 3 most relevant documents in each dataset.
# - Use cosine similarity between query embeddings and stored vector embe
web_results = databases["Web_data"].similarity_search_with_score(query_te
book_results = databases["Book_data"].similarity_search_with_score(query_

# Sort the results by similarity score in descending order
# - Higher scores indicate more relevant matches.
web_results = sorted(web_results, key=lambda x: x[1], reverse=True)
book_results = sorted(book_results, key=lambda x: x[1], reverse=True)
```

In [30]:
```
# Display the top 3 results from web data (Semantic chunking)
display(Markdown(f"\n **Top 3 Results from Web Data (Semantice chunking):
for i, (doc, score) in enumerate(web_results, start=1):
    display(Markdown(f"\n **Result {i}** (Similarity Score: {score:.4f})"
    print(textwrap.fill(doc.page_content, width=100))  # Wrap text for be
    print("-" * 80)  # Separator for clarity

display(Markdown(f"\n **Top 3 Results from Book Data (Page-based chunking
for i, (doc, score) in enumerate(book_results, start=1):
    display(Markdown(f"\n **Result {i}** (Similarity Score: {score:.4f})"
    print(textwrap.fill(doc.page_content, width=100))
    print("-" * 80)
```

Top 3 Results from Web Data (Semantice chunking):

Result 1 (Similarity Score: 0.4977)

```
Most people think they know what they are good at. They are usually wrong.
More often, people know
what they are not good at — and even then more people are wrong than righ
t. And yet, a person can
perform only from strength. One cannot build performance on weaknesses, le
t alone on something one
cannot do at all. Throughout history, people had little need to know their
strengths. A person was
born into a position and a line of work: The peasant’s son would also be a
peasant; the artisan’s
daughter, an artisan’s wife; and so on. But now people have choices.
--------------------------------------------------------------------------
--------
```

Result 2 (Similarity Score: 0.4974)

Most people think they know what they are good at. They are usually wrong. More often, people know
what they are not good at — and even then more people are wrong than righ t. And yet, a person can
perform only from strength. One cannot build performance on weaknesses, le t alone on something one
cannot do at all. Throughout history, people had little need to know their strengths. A person was
born into a position and a line of work: The peasant’s son would also be a peasant; the artisan’s
daughter, an artisan’s wife; and so on. But now people have choices.

Result 3 (Similarity Score: 0.4968)

Most people think they know what they are good at. They are usually wrong. More often, people know
what they are not good at — and even then more people are wrong than righ t. And yet, a person can
perform only from strength. One cannot build performance on weaknesses, le t alone on something one
cannot do at all. Throughout history, people had little need to know their strengths. A person was
born into a position and a line of work: The peasant’s son would also be a peasant; the artisan’s
daughter, an artisan’s wife; and so on. But now people have choices.

Top 3 Results from Book Data (Page-based chunking):

Result 1 (Similarity Score: 0.6161)

24 January Feedback: Key to Continuous Learning To know one’s strengths, t o know how to improve
them, and to know what one cannot do—are the keys to continuous learning. W henever a Jesuit priest
or a Calvinist pastor does anything of significance (for instance, making a key decision), he is
expected to write down what results he anticipates. Nine months later, he then feeds back from the
actual results to these anticipations. This very soon shows him what he di d well and what his
strengths are. It also shows him what he has to learn and what habits he h as to change. Finally it
shows him what he is not gifted for and cannot do well. I have followed th is method myself, now for
fifty years. It brings out what one’s strengths are—and this is the most i mportant thing an
individual can know about himself or herself. It brings out where improvem ent is needed and what
kind of improvement is needed. Finally, it brings out what an individual c annot do and therefore
should not even try to do. To know one’s strengths, to know how to improve them, and to know what
one cannot do—they are the keys to continuous learning. ACTION POINT: Li st your strengths and the
steps you are taking to improve them. Who knows you well enough to help id entify your strengths?
Drucker on Asia

Result 2 (Similarity Score: 0.6160)

24 January Feedback: Key to Continuous Learning To know one’s strengths, to know how to improve
them, and to know what one cannot do—are the keys to continuous learning. Whenever a Jesuit priest
or a Calvinist pastor does anything of significance (for instance, making a key decision), he is
expected to write down what results he anticipates. Nine months later, he then feeds back from the
actual results to these anticipations. This very soon shows him what he did well and what his
strengths are. It also shows him what he has to learn and what habits he has to change. Finally it
shows him what he is not gifted for and cannot do well. I have followed this method myself, now for
fifty years. It brings out what one’s strengths are—and this is the most important thing an
individual can know about himself or herself. It brings out where improvement is needed and what
kind of improvement is needed. Finally, it brings out what an individual cannot do and therefore
should not even try to do. To know one’s strengths, to know how to improve them, and to know what
one cannot do—they are the keys to continuous learning. ACTION POINT: List your strengths and the
steps you are taking to improve them. Who knows you well enough to help identify your strengths?
Drucker on Asia

Result 3 (Similarity Score: 0.6159)

24 January Feedback: Key to Continuous Learning To know one’s strengths, to know how to improve
them, and to know what one cannot do—are the keys to continuous learning. Whenever a Jesuit priest
or a Calvinist pastor does anything of significance (for instance, making a key decision), he is
expected to write down what results he anticipates. Nine months later, he then feeds back from the
actual results to these anticipations. This very soon shows him what he did well and what his
strengths are. It also shows him what he has to learn and what habits he has to change. Finally it
shows him what he is not gifted for and cannot do well. I have followed this method myself, now for
fifty years. It brings out what one’s strengths are—and this is the most important thing an
individual can know about himself or herself. It brings out where improvement is needed and what
kind of improvement is needed. Finally, it brings out what an individual cannot do and therefore
should not even try to do. To know one’s strengths, to know how to improve them, and to know what
one cannot do—they are the keys to continuous learning. ACTION POINT: List your strengths and the
steps you are taking to improve them. Who knows you well enough to help identify your strengths?
Drucker on Asia

Your Turn: Analyze the Search Results

Now, take some time to review the retrieved results.

- Compare the results from semantic chunking (web data) and page-based chunking (book data).
- Discuss any insights gained from the differences in retrieved content.