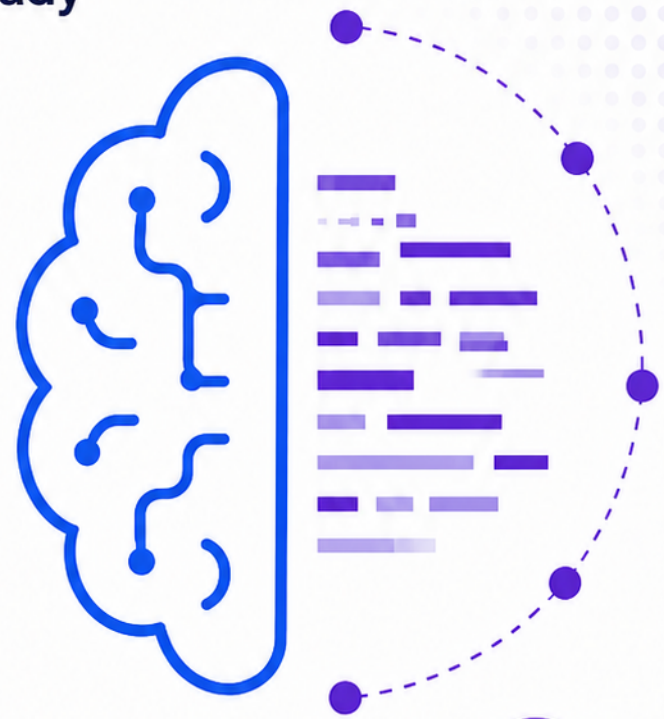
Production NLP with spaCy

A Comprehensive Guide to Building Production-Ready NLP Systems

UNDERSTAND
Language. Deeply.

TRAIN
Custom models.
Any task.

DEPLOY
Scale with confidence.



v3 COMPATIBLE

BEN MEGED

Production NLP with spaCy

A Comprehensive Guide to Building Production-Ready NLP Systems

Steve Publications

This book is available at https://leanpub.com/productionnlpwithspacy

This version was published on 2026-07-23

Contents

A Comprehensive Guide to Building Production-Ready NLP Systems

spaCy is the most practical, efficient, and production-oriented natural language processing library available for Python. This book takes you from complete beginner to advanced practitioner, covering everything from core concepts and linguistic analysis to training custom models, deploying in production, and solving real-world business problems across healthcare, finance, legal, and other domains. Every concept is explained with clear, in-depth reasoning followed by complete, working code examples that follow modern Python best practices. Whether you are a data scientist new to NLP or an experienced engineer building enterprise text-processing systems, this book gives you the knowledge and confidence to design, train, deploy, and maintain production-ready NLP pipelines using spaCy v3.x.

Introduction: Why spaCy, Why Now

Every day, organizations generate and consume enormous volumes of text: customer support tickets, medical records, financial reports, legal contracts, social media posts, product reviews, news articles. This text contains valuable information that can drive better decisions, automate workflows, and create competitive advantages. The challenge is that computers do not understand human language the way we do. They see sequences of characters and symbols without inherent meaning. Natural language processing bridges this gap, enabling machines to extract structure, semantics, and insight from the messy, ambiguous world of human text.

This introduction sets the stage for the journey ahead. We will explore what natural language processing is, why spaCy has become the library of choice for production NLP, how it compares to alternatives, and what you can accomplish with it by the end of this book.

What Is Natural Language Processing?

Natural language processing, or NLP, is a subfield of artificial intelligence focused on enabling computers to understand, interpret, and generate human language. It sits at the intersection of linguistics, computer science, and machine learning. NLP encompasses a wide range of tasks:

- **Tokenization**: Breaking text into meaningful units such as words or subwords.
- **Part-of-speech tagging**: Identifying whether each word is a noun, verb, adjective, or other grammatical category.
- **Dependency parsing**: Determining the grammatical relationships between words in a sentence.
- **Named entity recognition**: Finding and classifying real-world objects such as people, organizations, locations, dates, and monetary amounts.
- **Text classification**: Assigning documents to predefined categories such as spam versus not spam, or positive versus negative sentiment.

- **Similarity and embeddings**: Representing text as numerical vectors that capture semantic meaning, enabling comparison of how similar two pieces of text are.
- **Relation extraction**: Identifying relationships between entities, such as "Apple acquired Beats" expressing an acquisition relationship.
- **Question answering and dialogue systems**: Understanding questions and generating appropriate responses.

These tasks form the foundation for practical applications: search engines that understand intent, chatbots that handle customer inquiries, document processing systems that extract key information from contracts, recommendation systems that analyze product descriptions, and monitoring tools that track sentiment about brands or products across social media.

Why spaCy for Production NLP?

Many NLP libraries exist, each with different design philosophies. spaCy was created by Matthew Honnibal and Ines Montani and is developed by the company Explosion AI. What distinguishes spaCy from the start is its design goal: it was built from day one to be used in real products, not just research papers or classroom exercises [1].

Here are the core reasons spaCy has become the industry standard for production NLP:

Speed and efficiency. spaCy is written primarily in Cython, a language that compiles to C, giving it performance comparable to compiled languages while maintaining Python's ease of use. In official benchmarks on 10,000 Reddit comments, en_core_web_lg processes approximately 10,014 words per second on CPU and 14,954 words per second on GPU, compared to Stanza at 878 WPS (CPU) and Flair at 323 WPS (CPU) [30]. This order-of-magnitude speed advantage matters in production environments processing millions of documents or responding to user queries in milliseconds.

Accuracy with state-of-the-art models. spaCy ships with pretrained statistical models trained on large corpora using modern neural network architectures. For English, the transformer-based en_core_web_trf model achieves 90.19 F1 on NER, 98.13 percent accuracy on POS tagging, and 93.91 LAS (Labeled Attachment Score) for dependency parsing on the OntoNotes

5.0 corpus [31]. The CNN-based en_core_web_lg achieves approximately 85.4 F1 on NER using the same evaluation framework [30]. With transformer-based pipelines introduced in spaCy v3, you can leverage RoBERTa and other architectures within a spaCy pipeline, bringing accuracy to competitive state-of-the-art levels when required [4].

Production-oriented design. Unlike research-focused libraries that expose every possible algorithm and option, spaCy provides curated, opinionated defaults that work well out of the box. Its pipeline architecture makes it easy to combine multiple processing steps, disable components you do not need, and swap in custom components. The library includes built-in support for serialization, multiprocessing, and deployment patterns that matter when you move from prototype to production.

Extensibility. When the defaults are not enough, spaCy makes it straightforward to extend its capabilities. You can write custom pipeline components, add attributes to documents and tokens, integrate with other machine learning frameworks, and train models on your own domain-specific data. The spaCy Universe ecosystem hosts hundreds of community-contributed extensions for specialized tasks [5].

Multilingual support. spaCy supports tokenization for over seventy languages and provides full NLP pipelines (including tagging, parsing, and named entity recognition) for more than twenty languages including English, German, French, Spanish, Chinese, and Japanese [6]. This makes it practical for global applications that need to process text in multiple languages.

Strong documentation and community. spaCy is known for its clear, example-driven documentation. The official website provides comprehensive guides covering every aspect of the library. A large and active community contributes to discussions on GitHub, Stack Overflow, and other platforms, making it easier to find solutions to problems you encounter.

When spaCy Is Not the Right Tool

Before diving into comparisons, it is worth being explicit about when spaCy is not the best choice. No single library solves every NLP problem, and understanding spaCy's limitations helps you design better systems.

Consider alternatives when:

- **You need generative capabilities.** spaCy excels at analysis and extraction but cannot generate text, summarize documents in natural language, or answer open-ended questions. For these tasks, large language models (GPT-4, Claude, Llama) or dedicated generation frameworks are necessary. In practice, many production systems combine both: spaCy for efficient structured extraction and LLMs for generative tasks where their capabilities are genuinely needed.
- **Your primary need is research experimentation.** If you want to prototype novel algorithms, compare dozens of tokenization strategies, or experiment with cutting-edge linguistic theories, NLTK or Hugging Face may provide more flexibility. spaCy's opinionated design prioritizes production stability over research convenience.
- **You require support for very low-resource languages.** While spaCy supports 75+ languages for tokenization and 20+ for full pipelines, many languages have no pretrained models at all. For these languages, you may need to build from scratch using more flexible frameworks or use multilingual transformer models as a starting point.
- **You need nested entity recognition out of the box.** spaCy's default NER does not support nested entities (e.g., "University of California, Berkeley" where "Berkeley" is both part of an ORG and a separate GPE). While workarounds exist using custom components and span stores, libraries like BRAT or Flair handle nested annotations more naturally.
- **Your task requires reasoning over long documents.** spaCy processes text efficiently but does not maintain context across very long documents the way transformer-based models or retrieval-augmented generation systems do. For tasks requiring understanding of document-level relationships, consider combining spaCy with retrieval systems or long-context models.

Being honest about these limitations is important because the best NLP systems are often hybrid architectures that use the right tool for each subtask rather than forcing one library to do everything.

Comparing spaCy to Alternatives

Understanding where spaCy fits in the broader NLP ecosystem helps you make informed tool choices. Let us briefly compare spaCy to the most common alternatives:

NLTK (Natural Language Toolkit). NLTK is one of the oldest and most widely used NLP libraries, especially in academic settings and education. It provides implementations of many classic NLP algorithms, extensive corpora, and a modular design that lets you experiment with different approaches [7]. However, NLTK prioritizes flexibility and educational value over production performance. Its models are often older, its processing is slower due to pure-Python implementation, and it requires more boilerplate code to achieve the same results. NLTK remains excellent for learning NLP fundamentals and research prototyping, but spaCy is the better choice when you need speed, accuracy, and production readiness.

Stanza (Stanford NLP). Stanza is Stanford University's modern NLP library, offering strong multilingual support and competitive accuracy, particularly for languages where spaCy has fewer resources [8]. Stanza uses neural network models trained on Universal Dependencies corpora and provides consistent quality across many languages. However, it tends to be slower than spaCy and has a smaller ecosystem of extensions. If your primary need is high-quality processing for a language with limited spaCy support, Stanza is worth considering.

Hugging Face Transformers. The Hugging Face ecosystem has revolutionized NLP by making transformer models accessible to everyone. Libraries like transformers and sentence-transformers provide access to thousands of pretrained models for virtually every NLP task [9]. However, Hugging Face focuses on model access rather than end-to-end pipeline management. When you need a complete system that combines tokenization, parsing, named entity recognition, and custom components in a single optimized pipeline, spaCy provides better structure and tooling. In practice, many production systems use both: spaCy for efficient pipeline orchestration and Hugging Face models accessed through spaCy's transformer integration when you need the highest accuracy.

spaCy versus large language models. The rise of large language models such as GPT-4 and Claude has raised questions about whether traditional NLP libraries are still relevant. The answer is yes, for several reasons. First, LLMs are expensive and slow compared to specialized models like spaCy pipelines. If you need to process millions of documents or respond in real time, dedicated NLP pipelines are more practical. Second, spaCy provides deterministic, interpretable results, whereas LLM outputs can be unpredictable and hallucinate information. Third, you can combine both approaches: use spaCy for efficient

preprocessing and structured extraction, and call an LLM only when you need its generative capabilities or reasoning abilities.

What You Will Learn in This Book

This book is organized to take you progressively from foundational concepts to advanced production techniques. Here is the journey ahead:

Chapters 1 through 3 establish the foundation. You will learn how to install spaCy, understand its architecture, and work with core objects like documents, tokens, and spans. You will explore tokenization, lexical attributes, and text preprocessing techniques that form the basis for all NLP tasks.

Chapters 4 through 6 cover linguistic analysis. You will master part-of-speech tagging, lemmatization, morphological features, dependency parsing, and named entity recognition. These chapters explain not just how to use these features, but what they represent linguistically and how to apply them in real applications.

Chapters 7 and 8 focus on extraction and classification. You will learn rule-based matching using spaCy's Matcher, PhraseMatcher, and DependencyMatcher for deterministic pattern extraction. You will also explore text classification, word vectors, embeddings, and similarity analysis.

Chapter 9 introduces transformers and modern architectures, showing how to integrate BERT, RoBERTa, and other transformer models into spaCy pipelines for state-of-the-art accuracy when you need it.

Chapters 10 through 12 cover training and customization. You will learn how to prepare data, train custom named entity recognition models, fine-tune existing pipelines, extend spaCy with custom components, and use the configuration system for reproducible experiments.

Chapter 13 covers project workflows, showing how to organize, version, and share NLP projects professionally using spaCy's project CLI.

Chapter 14 addresses production deployment and optimization, including serving models via FastAPI, batch processing strategies, memory management, performance tuning, and monitoring.

Chapter 15 brings everything together with real-world applications across domains: information extraction, document processing, search enhancement,

chatbots, healthcare, finance, legal, and customer support. You will see how to integrate spaCy with pandas, scikit-learn, Streamlit, and other tools commonly used in data science and software engineering.

By the end of this book, you will be able to independently design and build production-ready NLP systems using spaCy for real business problems. You will understand the trade-offs between different approaches, know how to troubleshoot common issues, and have the confidence to tackle new NLP challenges as they arise.

How to Use This Book

This book is designed for self-study and practical reference. Each chapter builds on previous material, so reading sequentially is recommended, especially if you are new to NLP. However, once you are familiar with the basics, you can jump to specific chapters relevant to your current project.

Every concept includes complete, working Python code examples. These examples use realistic sample text rather than trivial placeholders, and they follow modern Python best practices including type hints, clear naming, and proper error handling. All code is written for spaCy v3.x and tested against the latest stable release.

You will need a working Python installation (version 3.9 or later) to run the examples. Each chapter specifies any additional dependencies required. We recommend using a virtual environment to isolate spaCy and its models from your system Python.

A Note on spaCy Versions

This book is written for spaCy v3.x, specifically targeting version 3.8 and later releases. spaCy v3 introduced significant changes compared to v2, including a new configuration system, transformer support, and updated training workflows. If you are upgrading from spaCy v2, refer to the official migration guide at spacy.io/usage/v3 for details on breaking changes.

The spaCy team follows semantic versioning, and minor updates within the 3.x series generally maintain backward compatibility. However, always check release notes when upgrading in production environments, as subtle behavioral changes can affect your pipelines.

This book assumes you are comfortable with Python programming but requires no prior experience with natural language processing or machine learning. We will explain NLP concepts as we encounter them, building up your understanding progressively. Let us begin the journey by getting spaCy installed and running your first program.

Chapter Summary

Natural language processing enables computers to understand and extract meaning from human text, powering applications from search engines to chatbots to document processing systems. spaCy has become the industry standard for production NLP due to its speed, accuracy, production-oriented design, extensibility, multilingual support, and strong documentation. While alternatives like NLTK, Stanza, and Hugging Face Transformers each have their strengths, spaCy provides the best balance of performance and practicality for building real-world NLP systems. Large language models complement rather than replace tools like spaCy in production environments. This book takes you from installation through advanced production deployment, with complete working code examples throughout, preparing you to build NLP systems that solve actual business problems.

Chapter 1: Getting Started with NLP and spaCy

Before we can build sophisticated NLP systems, we need to understand what natural language processing is, why spaCy is the right tool for production workloads, and how to set up a working environment. This chapter establishes the foundation everything else builds upon. By the end of this chapter, you will have spaCy installed, understand its place in the NLP ecosystem, and have run your first complete NLP program.

What Is Natural Language Processing?

Natural language processing is the field of artificial intelligence concerned with enabling computers to understand, interpret, manipulate, and generate human language. The term "natural language" distinguishes everyday human languages like English, Spanish, or Mandarin from formal programming languages like Python or SQL. Human language is messy, ambiguous, context-dependent, and full of exceptions to rules. Programming languages are precise, unambiguous, and deterministic. NLP sits in the difficult middle ground.

At a high level, NLP involves two broad categories of tasks: natural language understanding and natural language generation. Understanding tasks include parsing text to identify its structure, extracting entities and relationships, determining sentiment or intent, and classifying documents. Generation tasks include producing human-readable text from structured data, summarizing long documents, translating between languages, and generating responses in conversational systems.

To accomplish these tasks, NLP systems typically process text through a series of steps:

1. **Tokenization**: Splitting raw text into smaller units called tokens, usually words or subwords.
2. **Normalization**: Converting text to a standard form, such as lowercasing or removing punctuation.

3. **Lemmatization or stemming**: Reducing words to their base or root forms so that "running", "runs", and "ran" are treated as variants of the same word.
4. **Part-of-speech tagging**: Identifying the grammatical category of each token (noun, verb, adjective, etc.).
5. **Parsing**: Determining the syntactic structure of sentences and relationships between words.
6. **Semantic analysis**: Extracting meaning, identifying entities, and understanding context.

These steps form a pipeline: each stage builds on the output of previous stages to add increasingly sophisticated annotations to the text. spaCy implements this pipeline architecture as its core design principle, which we will explore in detail in the next chapter.

Real-world NLP applications span virtually every industry. In healthcare, NLP extracts diagnoses and medications from clinical notes. In finance, it analyzes earnings reports and detects fraud indicators. In legal technology, it reviews contracts for key clauses and obligations. In customer support, it routes tickets based on intent and extracts issues from complaints. In marketing, it measures brand sentiment across social media. The common thread is that all these applications start with the same fundamental NLP tasks that spaCy excels at.

Why spaCy for Production NLP?

When choosing an NLP library, you face trade-offs between ease of use, flexibility, performance, accuracy, and ecosystem maturity. Different libraries optimize for different points in this space. spaCy's design philosophy is explicit: it prioritizes production readiness above all else.

Here are the specific reasons spaCy has become the default choice for production NLP in industry:

Performance. spaCy is implemented primarily in Cython, a superset of Python that compiles to C code. This gives spaCy performance comparable to compiled languages while maintaining Python's readability and ecosystem. In practice, this means spaCy can process text at speeds of tens of thousands of words per second on a single CPU core, making it suitable for real-time

applications and large-scale batch processing. The library is also memory-efficient, using shared vocabularies and optimized data structures to minimize overhead.

Accuracy out of the box. spaCy ships with pretrained statistical models trained on large, high-quality corpora using modern neural network architectures. These models provide competitive accuracy on standard benchmarks without requiring any training on your part. For English, spaCy's models are trained on web text and achieve strong performance on named entity recognition, part-of-speech tagging, and dependency parsing tasks. When you need even higher accuracy, spaCy v3 supports transformer-based pipelines that leverage models like BERT and RoBERTa.

Opinionated defaults. Research-oriented NLP libraries often expose dozens of options and algorithms for each task, requiring deep expertise to choose the right combination. spaCy takes a different approach: it provides well-chosen defaults that work well for most use cases. You can customize everything if needed, but you do not have to. This reduces the barrier to entry while still supporting advanced use cases.

Pipeline architecture. spaCy processes text through a configurable pipeline of components, each responsible for a specific task. This design makes it easy to understand what is happening, disable components you do not need, add custom processing steps, and optimize performance by controlling the order of operations. The pipeline model also simplifies deployment, since your entire NLP system is encapsulated in a single loadable object.

Extensibility. When built-in features are insufficient, spaCy provides clean APIs for extending functionality. You can write custom pipeline components that integrate seamlessly with existing ones, add custom attributes to documents and tokens, register your own model architectures, and combine spaCy with other machine learning frameworks. The spaCy Universe ecosystem hosts hundreds of community-contributed extensions for specialized tasks.

Multilingual support. spaCy supports tokenization for over seventy languages and provides full NLP pipelines for more than twenty. This includes major world languages like English, German, French, Spanish, Chinese, Japanese, Arabic, and many others. Each language has its own tokenizer rules, morphological analysis, and pretrained models tuned to that language's characteristics.

Documentation and community. spaCy is known for its exceptionally

clear documentation. The official website provides comprehensive guides, API references, and tutorials with working code examples. A large and active community contributes to discussions on GitHub, Stack Overflow, and other platforms. The library is maintained by Explosion AI, a commercial company that invests in long-term development and support.

Production features. spaCy includes features that matter in production environments: efficient serialization for saving and loading models, built-in support for multiprocessing and batch processing, configuration management for reproducible experiments, project workflows for end-to-end pipeline management, and tools for evaluating model performance. These features reduce the gap between prototype and production.

Comparing spaCy to NLTK, Stanza, Hugging Face Transformers

To understand where spaCy fits in the NLP ecosystem, it helps to compare it with the most common alternatives. Each library has strengths and weaknesses that make it suitable for different scenarios.

NLTK (Natural Language Toolkit) is one of the oldest and most widely used NLP libraries, particularly in academic settings. NLTK provides implementations of many classic NLP algorithms, extensive text corpora, and a modular design that lets you experiment with different approaches. Its strength lies in education and research: if you want to understand how different tokenization algorithms work or compare multiple parsing strategies, NLTK is excellent. However, NLTK prioritizes flexibility over performance. It is implemented in pure Python, making it significantly slower than spaCy for most tasks. Its pretrained models are older and less accurate than spaCy's. For production systems where speed and accuracy matter, spaCy is the better choice.

Stanza is Stanford University's modern NLP library, offering neural network-based models trained on Universal Dependencies corpora. Stanza provides strong multilingual support and competitive accuracy, particularly for languages where spaCy has fewer resources. It is a good choice when you need high-quality processing for less common languages or when you specifically want Universal Dependencies annotations. However, Stanza tends to be slower than spaCy and has a smaller ecosystem of extensions and community resources.

Hugging Face Transformers has revolutionized NLP by making transformer models accessible to everyone. The transformers library provides access to thousands of pretrained models for virtually every NLP task, from text classification to question answering to text generation. When you need cutting-edge accuracy on a specific task, Hugging Face is often the best starting point. However, Hugging Face focuses on model access rather than end-to-end pipeline management. It does not provide the same level of integration between tokenization, parsing, named entity recognition, and other core NLP tasks that spaCy offers. In practice, many production systems use both: spaCy for efficient pipeline orchestration and core NLP tasks, with Hugging Face models accessed through spaCy's transformer integration when maximum accuracy is needed.

The following table summarizes the comparison:

Feature	**spaCy**	**NLTK**	**Stanza**	**Hugging Face Transformers**
Primary focus	Production NLP pipelines	Education and research	Multilingual NLP	Transformer model access
Performance	Very fast (Cython)	Slow (pure Python)	Moderate	Varies by model
Pretrained models	Yes, optimized	Limited, older	Yes, UD-based	Thousands available
Pipeline integration	Excellent	Manual composition	Good	Task-specific models
Multilingual support	70+ tokenization, 20+ full	Many languages	Many languages	Model-dependent
Learning curve	Moderate	Low to moderate	Moderate	High for advanced use

Feature	spaCy	NLTK	Stanza	Hugging Face Transformers
Best for	Production systems	Learning NLP basics	UD annotations, less common languages	State-of-the-art accuracy on specific tasks

In most production scenarios, spaCy is the right starting point. You can always integrate other libraries when you need capabilities spaCy does not provide directly.

Installation and Environment Setup

Let us get spaCy installed and ready to use. This section walks through the installation process step by step, covering different environments and options.

Prerequisites. spaCy requires Python 3.9 or later running on a 64-bit system. It supports Linux, macOS, and Windows. Before installing spaCy, ensure you have a working Python installation. You can check your Python version by running:

```
import sys
print(sys.version)
```

Using pip. The recommended way to install spaCy is via pip, Python's package manager. Open a terminal or command prompt and run:

```
pip install spacy
```

This installs the latest stable version of spaCy along with its core dependencies. After installation, verify it works by running:

```
import spacy
print(spacy.__version__)
```

You should see the version number, which will be 3.8 or later if you installed from the current release.

Installing language models. spaCy itself does not include pretrained language models. You must install them separately. Models are distributed as Python packages and can be installed using spaCy's download command. For English, the most commonly used model is en_core_web_sm, a small general-purpose model:

```
python -m spacy download en_core_web_sm
```

This downloads and installs the model, making it available for use in your code. Other English models include:

- en_core_web_md: Medium-sized model with word vectors
- en_core_web_lg: Large model with more extensive word vectors
- en_core_web_trf: Transformer-based model using RoBERTa for highest accuracy

You can list all available models for a language by visiting the spaCy models page at https://spacy.io/models. For example, German models are listed at https://spacy.io/models/de.

Using conda. If you prefer conda for environment management, spaCy is available through conda-forge:

```
conda install -c conda-forge spacy
```

After installation, download models the same way as with pip using the spaCy download command.

GPU support. For transformer-based models, GPU acceleration can significantly improve performance. spaCy supports GPU computation through CuPy. Install the appropriate variant for your CUDA version:

```
pip install spacy[cuda12x]
```

Replace cuda12x with the appropriate CUDA version suffix (cuda11x, cuda12x, etc.) matching your system. After installation, you can activate GPU processing in your code using spacy.prefer_gpu(). Note that transformer models require substantial GPU memory: at least 10 GB is recommended for comfortable operation.

Virtual environments. For production projects, always use a virtual environment to isolate spaCy and its dependencies from your system Python. This prevents version conflicts and makes it easier to reproduce your environment on other machines. Using venv:

```
python -m venv nlp-env
source nlp-env/bin/activate  # On Linux/macOS
# Or on Windows:
# nlp-env\Scripts\activate
pip install spacy
python -m spacy download en_core_web_sm
```

Common installation issues. If you encounter compilation errors during installation, you may need to install system-level build tools. On Ubuntu/Debian:

```
sudo apt-get install build-essential python3-dev
```

On macOS with Homebrew:

```
brew install gcc
```

On Windows, ensure you have Visual Studio Build Tools installed. If you see errors about incompatible model versions, make sure your spaCy version and model version match. You can check installed models with:

```
import spacy
print(spacy.util.get_installed_models())
```

Your First spaCy Program: End-to-End Example

Now that spaCy is installed, let us write a complete program that demonstrates its core capabilities. This example processes a realistic text and extracts multiple types of linguistic information, giving you a taste of what spaCy can do.

Create a file called first_program.py with the following code:

```
import spacy


def main() -> None:
    # Load the pretrained English model
    nlp = spacy.load("en_core_web_sm")

    # Sample text: a realistic business email snippet
    text = (
        "Apple Inc. announced today that CEO Tim Cook will visit Berlin,
        ↪  Germany "
        "next month to discuss a new partnership with Deutsche Telekom. The
        ↪  deal, "
        "worth approximately 500 million euros, is expected to expand Apple's "
        "presence in the European market."
    )

    # Process the text through the spaCy pipeline
    doc = nlp(text)

    print("=== TOKENS ===")
    for token in doc:
        print(f"{token.text:<15} POS={token.pos_:<6} Lemma={token.lemma_:<12}")

    print("\n=== NAMED ENTITIES ===")
    for ent in doc.ents:
        print(f"{ent.text:<25} Label={ent.label_}
        ↪  ({spacy.explain(ent.label_)})")

    print("\n=== SENTENCES ===")
    for i, sent in enumerate(doc.sents, 1):
        print(f"Sentence {i}: {sent.text}")

    print("\n=== KEYWORD EXTRACTION (Content Words) ===")
    content_words = [
        token.lemma_ for token in doc
        if token.pos_ in ("NOUN", "VERB", "ADJ") and not token.is_stop
    ]
    print(" ".join(content_words))
```

```

if __name__ == "__main__":
    main()
```

When you run this program, you will see output similar to the following:

```
=== TOKENS ===
Apple              POS=PROPN  Lemma=Apple
Inc.               POS=PROPN  Lemma=Inc.
announced          POS=VERB   Lemma=announce
today              POS=NOUN   Lemma=today
that               POS=SCONJ  Lemma=that
CEO                POS=NOUN   Lemma=CEO
Tim                POS=PROPN  Lemma=Tim
Cook               POS=PROPN  Lemma=Cook
will               POS=AUX    Lemma=will
visit              POS=VERB   Lemma=visit
Berlin             POS=PROPN  Lemma=Berlin
,                  POS=PUNCT  Lemma=,
Germany            POS=PROPN  Lemma=Germany
next               POS=ADJ    Lemma=next
month              POS=NOUN   Lemma=month
to                 POS=PART   Lemma=to
discuss            POS=VERB   Lemma=discuss
a                  POS=DET    Lemma=a
new                POS=ADJ    Lemma=new
partnership        POS=NOUN   Lemma=partnership
with               POS=ADP    Lemma=with
Deutsche           POS=PROPN  Lemma=Deutsche
Telekom            POS=PROPN  Lemma=Telekom
.                  POS=PUNCT  Lemma=.
The                POS=DET    Lemma=the
deal               POS=NOUN   Lemma=deal
,                  POS=PUNCT  Lemma=,
worth              POS=ADJ    Lemma=worth
approximately      POS=ADV    Lemma=approximately
500                POS=NUM    Lemma=500
million            POS=NUM    Lemma=million
euros              POS=NOUN   Lemma=euro
,                  POS=PUNCT  Lemma=,
is                 POS=AUX    Lemma=be
expected           POS=VERB   Lemma=expect
to                 POS=PART   Lemma=to
expand             POS=VERB   Lemma=expand
Apple's            POS=PROPN  Lemma=Apple
presence           POS=NOUN   Lemma=presence
in                 POS=ADP    Lemma=in
the                POS=DET    Lemma=the
```

```
European          POS=ADJ    Lemma=European
market            POS=NOUN   Lemma=market
.                 POS=PUNCT  Lemma=.

=== NAMED ENTITIES ===
Apple Inc.                Label=ORG (Companies, agencies, institutions)
today                     Label=DATE (Absolute or relative dates or periods)
Tim Cook                  Label=PERSON (People, including fictional)
Berlin                    Label=GPE (Countries, cities, states)
Germany                   Label=GPE (Countries, cities, states)
next month                Label=DATE (Absolute or relative dates or periods)
Deutsche Telekom          Label=ORG (Companies, agencies, institutions)
500 million euros         Label=MONEY (Monetary values, including unit)
European                  Label=NORP (Nationalities or religious or political
↪  groups)

=== SENTENCES ===
Sentence 1: Apple Inc. announced today that CEO Tim Cook will visit Berlin,
↪  Germany next month to discuss a new partnership with Deutsche Telekom.
Sentence 2: The deal, worth approximately 500 million euros, is expected to
↪  expand Apple's presence in the European market.

=== KEYWORD EXTRACTION (Content Words) ===
Apple Inc. announce today CEO Tim Cook visit Berlin Germany discuss new
↪  partnership Deutsche Telekom deal worth approximately 500 million euro
↪  expect expand Apple presence European market
```

Let us walk through what this program does:

1. **Loading the model.** The line nlp = spacy.load("en_core_web_sm") loads the pretrained English model. This creates a Language object that contains the tokenizer, vocabulary, and pipeline components (tagger, parser, named entity recognizer). Loading happens once at startup; the same nlp object can then process unlimited texts efficiently.
2. **Processing text.** Calling doc = nlp(text) runs the entire pipeline on the input text. spaCy first tokenizes the text into individual tokens, then each pipeline component adds its annotations: the tagger assigns part-of-speech tags, the parser determines syntactic dependencies and sentence boundaries, and the named entity recognizer identifies entities. The result is a Doc object containing all this information.
3. **Accessing tokens.** Iterating over doc yields Token objects, each representing one word or punctuation mark. Each token has attributes like text (the original string), pos_ (coarse part-of-speech tag), lemma_ (base form of the word), and many others that we will explore in detail in later chapters.

4. **Accessing entities.** doc.ents is a sequence of Span objects representing named entities. Each entity has a label indicating its type (ORG for organization, PERSON for person, etc.). The spacy.explain() function provides human-readable descriptions of entity labels.
5. **Accessing sentences.** doc.sents iterates over sentence-level spans, determined by the parser based on syntactic structure. This is more accurate than simple period-based splitting.
6. **Keyword extraction.** The final section demonstrates a practical application: extracting content words (nouns, verbs, adjectives that are not stopwords). This is a common preprocessing step for tasks like document summarization or search indexing.

This single program demonstrates multiple core spaCy capabilities: tokenization, part-of-speech tagging, lemmatization, named entity recognition, sentence segmentation, and stop word detection. Each of these topics will be explored in depth in subsequent chapters.

Understanding the spaCy Processing Flow

Before moving on, it is worth understanding at a high level what happens when you call nlp(text). The processing flow follows these steps:

1. **Tokenization.** The raw text string is split into tokens using language-specific rules. This creates a Doc object with the token texts but no linguistic annotations yet. Tokenization is fast and rule-based, not neural.
2. **Pipeline components.** The Doc passes through each component in the pipeline in order. For en_core_web_sm, the default pipeline includes:

    - tok2vec: Computes vector representations of tokens (used by other components)
    - tagger: Assigns fine-grained part-of-speech tags
    - parser: Determines syntactic dependencies and sentence boundaries
    - attribute_ruler: Applies rule-based attribute mappings
    - lemmatizer: Computes base forms of words using tags and rules
    - ner: Identifies named entities

3. **Annotation.** Each component modifies the Doc in place, adding its annotations. By the end of the pipeline, the Doc contains all available information about the text.

4. **Return.** The fully annotated Doc is returned to your code, where you can access any annotation through the appropriate attributes.

This pipeline architecture is central to how spaCy works and provides several advantages: components are modular and can be added, removed, or reordered; you can disable components you do not need for performance; custom components integrate seamlessly; and the entire pipeline can be serialized and loaded as a single unit. We will explore this architecture in detail in the next chapter.

Chapter Summary

Natural language processing enables computers to understand and extract meaning from human text through a series of processing steps including tokenization, part-of-speech tagging, parsing, and semantic analysis. spaCy has become the industry standard for production NLP due to its Cython-based performance, accurate pretrained models, opinionated defaults, modular pipeline architecture, extensibility, multilingual support, excellent documentation, and production-ready features. Compared to alternatives like NLTK (better for education), Stanza (strong multilingual UD support), and Hugging Face Transformers (access to cutting-edge models), spaCy provides the best balance of speed, accuracy, and practicality for building real-world NLP systems. Installation is straightforward via pip or conda, with language models installed separately using spaCy's download command. Your first spaCy program demonstrates core capabilities including tokenization, part-of-speech tagging, lemmatization, named entity recognition, and sentence segmentation through a simple but realistic example. Understanding the processing flow from tokenization through pipeline components to annotated documents provides the foundation for everything that follows in this book.

Chapter 2: Core Concepts: Language Models, Tokenization, and Document Objects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The spaCy Pipeline Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Loading and Understanding Language Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The Doc Object: Your Processed Document

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Tokens: The Atomic Units of Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Spans: Working with Text Ranges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Tokenization Mechanics and Edge Cases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 3: Lexical Analysis and Text Attributes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Token Attributes: orth, lemma, shape, length, and Beyond

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Special Character Detection (Punctuation, Whitespace, Quotes)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Case Normalization and Unicode Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Building Text Preprocessing Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Practical Applications: Data Cleaning and Normalization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Multilingual NLP with spaCy: Tokenization, Models, and Challenges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 4: Part-of-Speech Tagging and Morphological Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Understanding Parts of Speech

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Universal POS Tags vs Fine-Grained Tags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Lemmatization: Getting to Word Roots

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Morphological Features (Tense, Number, Gender, Case)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Using POS Information in Real Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 5: Dependency Parsing and Syntactic Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

What Is Dependency Grammar?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Reading Dependency Trees

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Core Relations: Subject, Object, Modifiers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Navigating the Tree Programmatically

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Advanced Traversal Patterns and Use Cases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 6: Named Entity Recognition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

What Is Named Entity Recognition?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Built-in Entity Types and Their Meaning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Extracting and Working with Entities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Evaluating NER Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Handling Ambiguity and Edge Cases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Customizing Entity Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 7: Rule-Based Matching and Pattern Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

When to Use Rules vs Machine Learning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The Matcher: Token-Level Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The PhraseMatcher: Fast Phrase Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The DependencyMatcher: Syntax-Aware Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Combining Matchers in Production Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 8: Text Classification, Similarity, and Embeddings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Trainable Text Classification with TextCategorizer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Binary vs Multi-Label Classification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Word Vectors and Embeddings Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Computing Similarity Between Texts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Practical Applications: Sentiment, Topic Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 9: Transformers and Modern Architecture Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Why Transformers Changed NLP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Using Transformer Models in spaCy Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

spaCy's Hugging Face Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Performance vs Accuracy Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Fine-Tuning Transformer-Based Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 10: Training and Fine-Tuning Custom Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Preparing Training Data for spaCy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Creating Annotation Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Measuring Inter-Annotator Agreement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Using Prodigy and Other Annotation Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Training Custom NER Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Fine-Tuning Existing Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Evaluation Metrics and Model Selection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Error Analysis: Understanding Model Failures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Active Learning Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Detecting Bias in Training Data and Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Active Learning with spaCy Training

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Common Training Pitfalls and Solutions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 11: Entity Linking and Relation Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

What Is Entity Linking?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

spaCy's Entity Ruler and Entity Linker

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Connecting Entities to Wikidata

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Relation Extraction Concepts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Building Knowledge Graphs from Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 12: Custom Pipeline Components, Extensions, and Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Creating Custom Pipeline Components

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Adding Extensions to Doc, Span, and Token

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The spaCy Configuration System (Config)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Building Reproducible Pipelines with Config Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Advanced Pipeline Composition Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 13: Project Workflows and Reproducibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Understanding spaCy Projects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Setting Up a Project with spacy project init

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Custom Recipes and Automation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Versioning and Sharing Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

CI/CD for NLP Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 14: Production Deployment and Performance Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Deployment Options: FastAPI, Flask, spaCy Serving

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Batch Processing Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Memory Optimization and Model Disassembly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Performance Profiling and Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Monitoring and Maintaining Production Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Advanced Data Drift Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter 15: Real-World Applications and Integration Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Information Extraction and Document Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Search Enhancement and Semantic Retrieval

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chatbots and Conversational NLP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Domain Applications: Healthcare, Finance, Legal, HR

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Integration with pandas, scikit-learn, Streamlit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Conclusion: Building NLP Systems That Last

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Architectural Principles for Production NLP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

The Evolving NLP Landscape

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

A Framework for Decision-Making

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Continuing Your Journey

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

Final Thoughts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/productionnlpwithspacy.