spaCy USAGE

Get started › Installation

Install spaCy

📖 LOOKING FOR THE OLD DOCS?

To help you make the transition from v2.x to v3.0, we've uploaded the old website to **v2.spacy.io**. To see what's changed and how to migrate, see the v3.0 guide.

Operating system

macOS / OSX | Windows | Linux

Platform

x86 | ARM / M1

Package manager

pip | conda | from source

Hardware

CPU | GPU | CUDA 11.2-11.x

Configuration

☐ virtual env ? ☐ train models ?

Trained pipelines

☐ Catalan ☐ Chinese ☐ Croatian ☐ Danish ☐ Dutch ☐ English ☐ Finnish
☐ French ☐ German ☐ Greek ☐ Italian ☐ Japanese ☐ Korean ☐ Lithuanian
☐ Macedonian ☐ Multi-language ☐ Norwegian Bokmål ☐ Polish ☐ Portuguese
☐ Romanian ☐ Russian ☐ Slovenian ☐ Spanish ☐ Swedish ☐ Ukrainian

Select pipeline for

spaCy

```
# Note M1 GPU support is experimental, see Thinc issue #792
$ conda install -c conda-forge spacy
$ conda install -c conda-forge cupy
$ python -m spacy download en_core_web_sm
$ python -m spacy download pt_core_news_sm
$ python -m spacy download es_core_news_sm
```

Installation instructions

spaCy is compatible with **64-bit CPython 3.7+** and runs on **Unix/Linux**, **macOS/OS X** and **Windows**. The latest spaCy releases are available over pip and conda.

pip

Using pip, spaCy releases are available as source packages and binary wheels. Before you install spaCy and its dependencies, make sure that your `pip`, `setuptools` and `wheel` are up to date.

DOWNLOAD PIPELINES

After installation you typically want to download a trained pipeline. For more info and available packages, see the models directory.

```
$ python -m spacy download en_core_web_sm
>>> import spacy
>>> nlp = spacy.load("en_core_web_sm")
```

```
$ pip install -U pip setuptools wheel
$ pip install -U spacy
```

spaCy

```
$ python -m venv .env
$ source .env/bin/activate
$ pip install -U pip setuptools wheel
$ pip install -U spacy
```

spaCy also lets you install extra dependencies by specifying the following keywords in brackets, e.g. `spacy[ja]` or `spacy[lookups,transformers]` (with multiple comma-separated extras). See the `[options.extras_require]` section in spaCy's `setup.cfg` </> for details on what's included.

EXAMPLE

```
$ pip install spacy[lookups,transformers]
```

NAME	DESCRIPTION
`lookups`	Install `spacy-lookups-data` </> for data tables for lemmatization and lexeme normalization. The data is serialized with trained pipelines, so you only need this package if you want to train your own models.
`transformers`	Install `spacy-transformers` </> . The package will be installed automatically when you install a transformer-based pipeline.
`cuda` , ...	Install spaCy with GPU support provided by CuPy for your given CUDA version. See the GPU installation instructions for details and options.
`apple`	Install `thinc-apple-ops` </> to improve performance on an Apple M1.
`ja` , `ko` , `th`	Install additional dependencies required for tokenization for the languages.

conda

Thanks to our great community, we've been able to re-add conda support. You can also install spaCy via `conda-forge` :

spaCy

For the feedstock including the build recipe and configuration, check out this repository </> . Note that we currently don't publish any pre-releases on conda.

Upgrading spaCy

UPGRADING FROM V2 TO V3

Although we've tried to keep breaking changes to a minimum, upgrading from spaCy v2.x to v3.x may still require some changes to your code base. For details see the sections on backwards incompatibilities and migrating. Also remember to download the new trained pipelines, and retrain your own pipelines.

When updating to a newer version of spaCy, it's generally recommended to start with a clean virtual environment. If you're upgrading to a new major version, make sure you have the latest **compatible trained pipelines** installed, and that there are no old and incompatible packages left over in your environment, as this can often lead to unexpected results and errors. If you've trained your own models, keep in mind that your train and runtime inputs must match. This means you'll have to **retrain your pipelines** with the new version.

spaCy also provides a `validate` command, which lets you verify that all installed pipeline packages are compatible with your spaCy version. If incompatible packages are found, tips and installation instructions are printed. It's recommended to run the command with `python -m` to make sure you're executing the correct version of spaCy.

```
$ pip install -U spacy
$ python -m spacy validate
```

Run spaCy with GPU

As of v2.0, spaCy comes with neural network models that are implemented in our machine learning library, Thinc. For GPU support, we've been grateful to use the work of Chainer's CuPy module, which provides a numpy-compatible interface for GPU arrays.

spaCy

using the more explicit specifier allows CuPy to be installed via wheel, saving some compilation time. The specifiers should install `cupy`.

```
$ pip install -U spacy[cuda113]
```

Once you have a GPU-enabled installation, the best way to activate it is to call `spacy.prefer_gpu` or `spacy.require_gpu()` somewhere in your script before any pipelines have been loaded. `require_gpu` will raise an error if no GPU is available.

```
import spacy

spacy.prefer_gpu()
nlp = spacy.load("en_core_web_sm")
```

Compile from source

The other way to install spaCy is to clone its GitHub repository and build it from source. That is the common way if you want to make changes to the code base. You'll need to make sure that you have a development environment consisting of a Python distribution including header files, a compiler, pip and git installed. The compiler part is the trickiest. How to do that depends on your system. See notes on Ubuntu, macOS / OS X and Windows for details.

```
$ python -m pip install -U pip setuptools wheel # install/update build tools
$ git clone https://github.com/explosion/spaCy  # clone spaCy
$ cd spaCy                                      # navigate into dir
$ python -m venv .env                           # create environment in .env
$ source .env/bin/activate                      # activate virtual env
$ pip install -r requirements.txt               # install requirements
$ pip install --no-build-isolation --editable . # compile and install spaCy
```

To install with extras:

spaCy

How to install compilers and related build tools:

- **Ubuntu:** Install system-level dependencies via `apt-get`: `sudo apt-get install build-essential python-dev git`
- **macOS / OS X:** Install a recent version of XCode, including the so-called "Command Line Tools". macOS and OS X ship with Python and Git preinstalled.
- **Windows:** Install a version of the Visual C++ Build Tools or Visual Studio Express that matches the version that was used to compile your Python interpreter.

Using build constraints when compiling from source

If you install spaCy from source or with `pip` for platforms where there are not binary wheels on PyPI, you may need to use build constraints if any package in your environment requires an older version of `numpy`.

If `numpy` gets downgraded from the most recent release at any point after you've compiled `spacy`, you might see an error that looks like this:

```
numpy.ndarray size changed, may indicate binary incompatibility.
```

To fix this, create a new virtual environment and install `spacy` and all of its dependencies using build constraints. Build constraints specify an older version of `numpy` that is only used while compiling `spacy`, and then your runtime environment can use any newer version of `numpy` and still be compatible. In addition, use `--no-cache-dir` to ignore any previously cached wheels so that all relevant packages are recompiled from scratch:

```
PIP_CONSTRAINT=https://raw.githubusercontent.com/explosion/spacy/master/build-co
pip install spacy --no-cache-dir
```

Our build constraints currently specify the oldest supported `numpy` available on PyPI for `x86_64` and `aarch64`. Depending on your platform and environment, you may want to customize the specific versions of `numpy`. For other platforms, you can have a look at SciPy's

spaCy

(*Warning*: don't use `pip install -c constraints.txt` instead of `PIP_CONSTRAINT`, since this isn't applied to the isolated build environments.)

Additional options for developers

Some additional options may be useful for spaCy developers who are editing the source code and recompiling frequently.

- Install in editable mode. Changes to `.py` files will be reflected as soon as the files are saved, but edits to Cython files (`.pxd`, `.pyx`) will require the `pip install` command below to be run again. Before installing in editable mode, be sure you have removed any previous installs with `pip uninstall spacy`, which you may need to run multiple times to remove all traces of earlier installs.

```
$ pip install -r requirements.txt
$ pip install --no-build-isolation --editable .
```

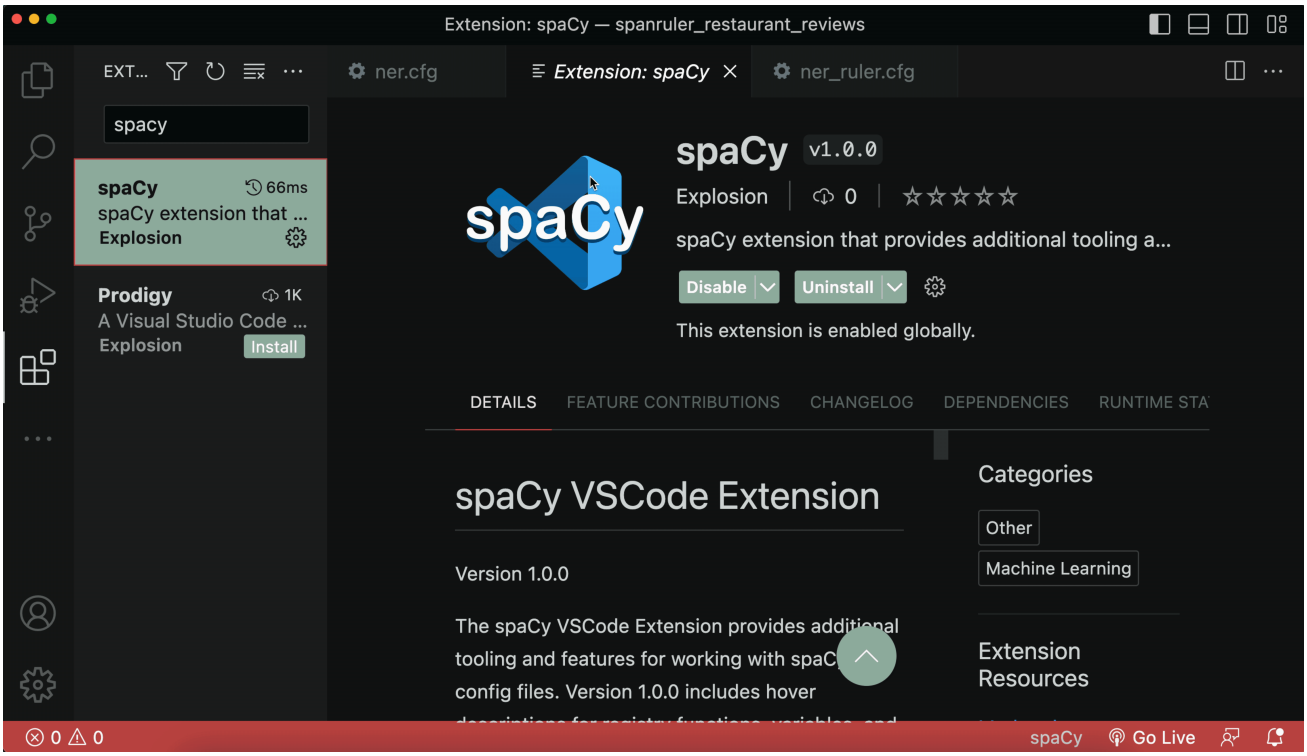
- Build in parallel. Starting in v3.4.0, you can specify the number of build jobs with the environment variable `SPACY_NUM_BUILD_JOBS`:

```
$ pip install -r requirements.txt
$ SPACY_NUM_BUILD_JOBS=4 pip install --no-build-isolation --editable .
```

- For editable mode and parallel builds with `python setup.py` instead of `pip` (no longer recommended):

```
$ pip install -r requirements.txt
$ python setup.py build_ext --inplace -j 4
$ python setup.py develop
```

spaCy



The spaCy VSCode Extension </> provides additional tooling and features for working with spaCy's config files. Version 1.0.0 includes hover descriptions for registry functions, variables, and section names within the config as an installable extension.

1. Install a supported version of Python on your system (`>=3.7`)
2. Install the Python Extension for Visual Studio Code
3. Create a virtual python environment
4. Install all python requirements (`spaCy >= 3.4.0` & `pygls >= 1.0.0`)
5. Install spaCy extension for Visual Studio Code
6. Select your python environment
7. You are ready to work with `.cfg` files in spaCy!

Building an executable

The spaCy repository includes a `Makefile` </> that builds an executable zip file using `pex` </> (**P**ython **Ex**ecutable). The executable includes spaCy and all its package dependencies and only requires the system Python at runtime. Building an executable `.pex` file is often the most convenient way to deploy spaCy, as it lets you separate the build from the deployment process.

spaCy

To use a `.pex` file, just replace `python` with the path to the file when you execute your code or CLI commands. This is equivalent to running Python in a virtual environment with spaCy installed.

```
$ ./spacy.pex my_script.py
$ ./spacy.pex -m spacy info
```

```
$ git clone https://github.com/explosion/spaCy
$ cd spaCy
$ make
```

You can configure the build process with the following environment variables:

VARIABLE	DESCRIPTION
`SPACY_EXTRAS`	Additional Python packages to install alongside spaCy with optional version specifications. Should be a string that can be passed to `pip install`. See `Makefile` </> for defaults.
`PYVER`	The Python version to build against. This version needs to be available on your build and runtime machines. Defaults to `3.8`.
`WHEELHOUSE`	Directory to store the wheel files during compilation. Defaults to `./wheelhouse`.

Run tests

spaCy comes with an extensive test suite </>. In order to run the tests, you'll usually want to clone the repository </> and build spaCy from source. This will also install the required development dependencies and test utilities defined in the `requirements.txt`.

Alternatively, you can find out where spaCy is installed and run `pytest` on that directory. Don't forget to also install the test utilities via spaCy's `requirements.txt` </>:

```
$ python -c "import os; import spacy; print(os.path.dirname(spacy.__file__))"
$ pip install -r path/to/requirements.txt
```

spaCy

Calling `pytest` on the spaCy directory will run only the basic tests. The flag `--slow` is optional and enables additional tests that take longer.

```
$ python -m pip install -U pytest                # update pytest
$ python -m pytest --pyargs spacy                # basic tests
$ python -m pytest --pyargs spacy --slow         # basic and slow tests
```

Troubleshooting guide

This section collects some of the most common errors you may come across when installing, loading and using spaCy, as well as their solutions. Also see the Discussions FAQ Thread </> , which is updated more frequently and covers more transitory issues.

HELP US IMPROVE THIS GUIDE

Did you come across a problem like the ones listed here and want to share the solution? You can find the “Suggest edits” button at the bottom of this page that points you to the source. We always appreciate pull requests !

No compatible model found

Import error: No module named spacy

Import error: No module named [name]

Command not found: spacy

'module' object has no attribute 'load'

spaCy

Unhashable type: 'list'

Changelog

Stable Releases

spaCy

2025-05-23	release-v3.8.7	Python 3.13 support, Cython 3, centralize registry entries
2025-05-19	release-v3.8.6	Restore wheels, remove Python 3.13 compatibility
2024-12-11	release-v3.8.3	Improve memory zone stability
2024-10-01	release-v3.8.2	Memory management for persistent services, numpy 2.0 support
2024-06-05	v3.7.5	Download sanitization, Typer compatibility, and a bugfix for linking gold entities
2024-02-15	v3.7.4	New textcat layers and fo/nn language extensions
2023-10-16	v3.7.2	Fixes for APIs and requirements
2023-10-05	v3.7.1	Bug fix for spacy.cli module loading
2023-10-02	v3.7.0	Trained pipelines using Curated Transformers and support for Python 3.12
2023-08-08	v3.6.1	Support for Pydantic v2, find-function CLI and more
2023-07-07	v3.6.0	New span finder component and pipelines for Slovenian
2023-06-28	v3.5.4	Bug fixes for overrides with registered functions and sourced components with listeners
2023-05-25	v3.3.3	Bug fixes for Pydantic and pip
2023-05-25	v3.2.6	Bug fixes for Pydantic and pip
2023-05-15	v3.5.3	Speed improvements, bug fixes and more
2023-04-12	v3.5.2	Pretraining improvements, bug fixes for spans and spancat and more
2023-03-10	v3.5.1	spancat for multi-class labeling, fixes for textcat+transformers and more
2023-01-20	v3.5.0	New CLI commands, language updates, bug fixes and much more
2022-12-16	v3.0.9	Bug fixes and future NumPy compatibility
2022-12-16	v2.3.9	Compatibility with NumPy v1.24+

spaCy

2022-12-16	`v3.2.5`	Bug fixes and future NumPy compatibility
2022-12-16	`v3.1.7`	Bug fixes and future NumPy compatibility
2022-11-10	`v3.4.3`	Extended Typer support and bug fixes
2022-10-20	`v3.4.2`	Latin and Luganda support, Python 3.11 wheels and more
2022-10-19	`v2.3.8`	Updates for Python 3.10 and 3.11
2022-07-26	`v3.4.1`	Fix compatibility with CuPy v9.x

Pre-Releases

Pre-releases include alpha and beta versions, as well as release candidates. They are not intended for production use. You can download spaCy pre-releases via the `spacy-nightly` package on pip.

`prerelease-v3.8.0.dev0` `prerelease-v3.7.6a`

SUGGEST EDITS

READ NEXT
Models & Languages →

spaCy

Your email SIGN UP